

GC28-1351-0  
File No. S370-36

**Program Product**

**MVS/Extended  
Architecture  
JCL User's Guide**

**MVS/System Product:**

**JES2 Version 2 5740-XC6  
JES3 Version 2 5665-291**

**IBM**

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

**MVS/System Product - JES2 Version 2 Release 1.2 (program number 5740-XC6)**

**MVS/System Product - JES3 Version 2 Release 1.2 (program number 5665-291)**

**MVS/Extended Architecture Data Facility Product (DFP) Release 1.2 (program number 5665-284)**

**Resource Access Control Facility (RACF) Version 1 Release 6 and later (program number 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 this new book has been improved and contains maintenance changes, installations using Version 2 Release 1.2 should use this book with its companion book, CG28-1352, even though the former book, GC28-1148-2 or GC28-1148-3, also reflects Version 2 Release 1.2.

### **First Edition (May, 1985)**

This book and *MVS/Extended Architecture JCL Reference*, GC28-1352-0, form a major revision of, and obsolete:

GC28-1148-2 with Technical Newsletter GN28-1013  
GC28-1148-3

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. The book, *MVS/Extended Architecture JCL*, GC28-1148-1, applies to Version 2 Release 1.1 and may now be ordered using the temporary order number GQ28-1148.

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 921-2, 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 the job control tasks needed to enter jobs into the operating system, control the system's processing of jobs, and request the resources needed to run jobs. To perform the tasks, programmers code job control statements. This publication describes how to use these statements, which consist of:

- Job control language (JCL) statements
- Job entry subsystem 2 (JES2) control statements
- Job entry subsystem 3 (JES3) control statements

This publication is designed as a user's guide, to be used when deciding how to perform job control tasks. It does not describe how to code the statements. For an introduction to the statements and for coding information, see the companion book, *MVS/Extended Architecture JCL Reference*, GC28-1352.

## Who Should Use This Publication

This book is needed by system and application programmers who enter programs into the operating system. Those using this book should understand the concepts of job management and data management.

## Information in This Publication

### Part 1. Introduction

**Chapter 1. Job Control Statements:** This chapter introduces the job control statements.

**Chapter 2. Job Control:** This chapter defines jobs, steps, input streams, and cataloged and in-stream procedures. It gives an overview of entering and processing jobs and requesting resources.

**Chapter 3. Job Control Tasks:** This chapter contains charts of job control tasks and the statements and parameters that can be used to perform the tasks. These charts indicate the organization for this book; the listed tasks are described in the rest of the book in the same order as in the charts.

## **Part 2. Tasks for Entering Jobs**

This part discusses the tasks for entering jobs into the system. Each chapter in this part describes a major task. The chapters are:

- Chapter 4. Identification
- Chapter 5. Execution
- Chapter 6. Job Input Control
- Chapter 7. Communication
- Chapter 8. Protection
- Chapter 9. Resource Control

## **Part 3. Tasks for Processing Jobs**

This part discusses the tasks for controlling the processing of jobs in the system. The chapters are:

- Chapter 10. Processing Control
- Chapter 11. Performance Control

## **Part 4. Tasks for Requesting Data Set Resources**

This part discusses the tasks for requesting data set resources. The chapters are:

- Chapter 12. Identification
- Chapter 13. Description
- Chapter 14. Protection
- Chapter 15. Allocation
- Chapter 16. Processing Control
- Chapter 17. End Processing

## **Part 5. Tasks for Requesting Sysout Data Set Resources**

This part discusses the tasks for requesting sysout data set resources. The sysout data sets are the output data sets that are processed by JES2 or JES3. The chapters are:

- Chapter 18. Identification
- Chapter 19. Description
- Chapter 20. Performance Control
- Chapter 21. Processing Control
- Chapter 22. End Processing
- Chapter 23. Output Destination
- Chapter 24. Output Formatting
- Chapter 25. Output Limiting

## **Part 6. Examples**

This part contains examples to show how to use JCL.

## Appendixes

The book contains three appendixes that detail the use of three types of data sets that have special requirements. The appendixes are:

- Appendix A. Indexed Sequential Data Sets
- Appendix B. Generation Data Sets
- Appendix C. VSAM Data Sets

## 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

*MVS/Extended Architecture System Programming Library: System Modifications*, GC28-1152.

*MVS/Extended Architecture Supervisor Services and Macro Instructions*, GC28-1154.

*MVS/Extended Architecture System Programming Library: System Macros and Facilities*, Volumes 1 and 2, GC28-1150 and GC28-1151.

*MVS/Extended Architecture Operations: System Commands*, GC28-1206.

*MVS/Extended Architecture System Programming Library: Initialization and Tuning*, GC28-1149.

*MVS/Extended Architecture System Programming Library: User Exits*, GC28-1147.

*MVS/Extended Architecture Diagnostic Techniques*, LY28-1199.

*MVS/Extended Architecture Debugging Handbook*, Volumes 1 through 5, LC28-1164 through LC28-1168.

*MVS/Extended Architecture Installation: System Generation*, GC26-4009.

### Data Facility Product

*MVS/Extended Architecture System-Data Administration*, GC26-4010.

*MVS/Extended Architecture Data Administration Guide*, GC26-4013.

*MVS/Extended Architecture Integrated Catalog Administration: Access Method Services Reference*, GC26-4019.

*MVS/Extended Architecture VSAM Catalog Administration: Access Method Services Reference*, GC26-4075.

*MVS/Extended Architecture Checkpoint/Restart User's Guide*, GC26-4012.

*MVS/Extended Architecture Data Administration: Utilities*, GC26-4018.

*MVS/Extended Architecture Magnetic Tape Labels and File Structure Administration*, GC26-4003.

*MVS/Extended Architecture VSAM Administration Guide*, GC26-4015.

*Advanced Communications Function for VTAM Version 2 Programming*, SC27-0611.

*Advanced Communications Function for TCAM, Version 2 Installation Reference*, SC30-3133.

## **JES2**

*MVS/Extended Architecture System Programming Library: JES2 Initialization and Tuning, SC23-0065.*

*MVS/Extended Architecture Operations: JES2 Commands, SC23-0064.*

## **JES3**

*MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning, SC23-0059.*

*MVS/Extended Architecture Operations: JES3 Commands, SC23-0063.*

*MVS/Extended Architecture JES3 Diagnosis, LC28-1370.*

*MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros, LC28-1372.*

## **Programs**

*MVS/Extended Architecture Interactive Problem Control System (IPCS) Guide and Reference, GC28-1297.*

*OS/VS Mass Storage System (MSS) Services General Information, GC35-0016.*

*Resource Access Control Facility (RACF) General Information Manual, GC28-0722.*

*MVS/Extended Architecture System Programming Library: Service Aids, GC28-1159.*

*MVS/Extended Architecture System Programming Library: System Management Facilities (SMF), GC28-1153.*

## **Hardware**

*Print Management Facility User's Guide and Reference, SH35-0059.*

*IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide, SH35-0061.*

*2821 Control Unit Component Description, GA24-3312.*

*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.*

*IBM 3340 Disk/Storage - Fixed Head Feature User's Guide, GA26-1632.*

# Contents

## Part 1. Introduction

### Chapter 1. Job Control Statements 1-1

### Chapter 2. Job Control 2-1

Entering Jobs 2-1

Processing Jobs 2-4

Requesting Resources 2-4

### Chapter 3. Job Control Tasks 3-1

## Part 2. Tasks for Entering Jobs

### Chapter 4. Identification 4-1

Identification of Job 4-2

Identification of Step 4-2

Identification of Procedure 4-2

Identification of Account 4-3

Identification of Programmer 4-4

### Chapter 5. Execution 5-1

Execution of Program 5-1

Execution of Procedure 5-2

Execution when Restarting and with Checkpointing 5-2

Deadline or Periodic Execution in a JES3 System 5-4

Execution when Dependent on Other Jobs in a JES3 System 5-5

Execution at Remote Node 5-7

### Chapter 6. Job Input Control 6-1

Job Input Control by Holding Job Entrance 6-1

Job Input Control by Holding Local Input Reader in a JES3 System 6-2

Job Input Control by Copying Input Stream in a JES2 System 6-3

Job Input Control from Remote Work Station 6-3

### Chapter 7. Communication 7-1

Communication from JCL to System 7-2

Communication from JCL to Operator 7-2

Communication from JCL to Programmer 7-2

Communication from JCL to Program 7-3

Communication from System to Operator 7-3

Communication from System to Time Sharing Userid 7-4

Communication from Time Sharing Userid to a JES3 System 7-5

Communication from Functional Subsystem to Programmer 7-6  
Communication through Job Log 7-6

**Chapter 8. Protection 8-1**  
Protection through RACF 8-1

**Chapter 9. Resource Control 9-1**  
Resource Control of Program Library 9-1  
Resource Control of Procedure Library 9-5  
Resource Control of Address Space 9-6  
Resource Control of the Processor 9-8  
Resource Control of Spool Partitions in a JES3 System 9-10

### **Part 3. Tasks for Processing Jobs**

**Chapter 10. Processing Control 10-1**  
Processing Control by Terminating Execution 10-2  
Processing Control by Timing Execution 10-10  
Processing Control for Testing 10-12

**Chapter 11. Performance Control 11-1**  
Performance Control by Job Class Assignment 11-2  
Performance Control by Selection Priority 11-3  
Performance Control by Dispatching Priority 11-4  
Performance Control by Performance Group Assignment 11-5  
Performance Control by I/O-to-Processing Ratio in a JES3 System 11-5

### **Part 4. Tasks for Requesting Data Set Resources**

**Chapter 12. Identification 12-1**  
Identification of Data Set 12-2  
Identification of In-Stream Data Set 12-5  
Identification of Data Set on 3540 Diskette Input/Output Unit 12-6  
Identification through Catalog 12-7  
Identification through Label 12-8  
Identification by Location on Tape 12-10  
Identification as TCAM Message Data Set 12-10  
Identification as Data Set from or to Terminal 12-11

**Chapter 13. Description 13-1**  
Description of Status 13-1  
Description of Data Attributes 13-5

**Chapter 14. Protection 14-1**  
Protection through RACF 14-1  
Protection for ISO/ANSI/FIPS Version 3 Tapes 14-2  
Protection by Passwords 14-2  
Protection of Access to BSAM or BDAM Data Sets 14-3

**Chapter 15. Allocation 15-1**  
Allocation of Device 15-1  
Allocation of Volume 15-15  
Allocation of Direct Access Space 15-21



Allocation of Virtual I/O	15-24
Allocation with Deferred Volume Mounting	15-27
Allocation with Volume Premounting in a JES2 System	15-27
Dynamic Allocation	15-28

<b>Chapter 16. Processing Control</b>	16-1
Processing Control by Suppressing Processing	16-1
Processing Control by Postponing Specification	16-2
Processing Control with Checkpointing	16-4
Processing Control by Subsystem	16-5
Processing Control by TCAM Job or Task	16-6

<b>Chapter 17. End Processing</b>	17-1
Deallocation End Processing	17-1
Disposition End Processing of Data Set	17-2
Release of Unused Direct Access Space in End Processing	17-9
Disposition End Processing of Volume	17-10

## **Part 5. Tasks for Requesting Sysout Data Set Resources**

<b>Chapter 18. Identification</b>	18-1
Identification as a Sysout Data Set	18-1
Identification of Output Class	18-2
Identification of Data Set on 3540 Diskette Input/Output Unit	18-2

<b>Chapter 19. Description</b>	19-1
Description of Data Attributes	19-1

<b>Chapter 20. Performance Control</b>	20-1
Performance Control by Queue Selection	20-1

<b>Chapter 21. Processing Control</b>	21-1
Processing Control with Additional Parameters	21-2
Processing Control with Other Data Sets	21-5
Processing Control by External Writer	21-7
Processing Control by Mode	21-7
Processing Control by Holding	21-8
Processing Control by Suppressing Output	21-9
Processing Control with Checkpointing	21-10
Processing Control by Print Services Facility	21-11

<b>Chapter 22. End Processing</b>	22-1
Deallocation End Processing	22-1

<b>Chapter 23. Destination Control</b>	23-1
Destination Control to Local or Remote Device or to Another Node	23-1
Destination Control to Another Processor in a JES3 System	23-4
Destination Control to Internal Reader	23-4
Destination Control to Terminal	23-6

<b>Chapter 24. Output Formatting</b>	24-1
Output Formatting to Any Printer	24-2
Output Formatting to 3800 Printing Subsystem	24-3
Output Formatting to 3211 Printer with Indexing Feature in a JES2 System	24-4

Output Formatting to Punch 24-5  
Output Formatting of Dumps on 3800 Printing Subsystem 24-6

**Chapter 25. Output Limiting** 25-1  
Output Limiting 25-1

## **Part 6. Examples**

**Chapter 26. Assemble, Linkedit, and Go** 26-1

**Chapter 27. Multiple Output** 27-1

**Chapter 28. Obtaining Output in a JES2 System** 28-1

**Chapter 29. Obtaining Output in a JES3 System** 29-1

**Chapter 30. Identifying Data Sets to the System** 30-1

## **Appendixes**

**Appendix A. Indexed Sequential Data Sets** A-1

**Appendix B. Generation Data Sets** B-1

**Appendix C. VSAM Data Sets** C-1

**Index** X-1

## Figures

- 1-1. Job Control Statements 1-1
- 2-1. Jobs and Job Steps 2-1
- 2-2. Job Boundaries in the Input Stream 2-2
- 2-3. In-Stream and Cataloged Procedures 2-3
- 2-4. JES Control Statements in Jobs 2-4
- 3-1. Tasks for Entering Jobs 3-2
- 3-2. Tasks for Processing Jobs 3-5
- 3-3. Tasks for Requesting Data Set Resources 3-6
- 3-4. Tasks for Requesting Sysout Data Set Resources 3-9
- 4-1. Identification Task for Entering Jobs 4-1
- 5-1. Execution Task for Entering Jobs 5-1
- 6-1. Input Control Task for Entering Jobs 6-1
- 7-1. Communication Task for Entering Jobs 7-1
- 8-1. Protection Task for Entering Jobs 8-1
- 9-1. Resource Control Task for Entering Jobs 9-1
- 10-1. Processing Control Task for Processing Jobs 10-1
- 11-1. Performance Control Task for Processing Jobs 11-1
- 12-1. Identification Task for Requesting Data Set Resources 12-1
- 13-1. Description Task for Requesting Data Set Resources 13-1
- 13-2. Data Set Integrity Processing 13-4
- 14-1. Protection Task for Requesting Data Set Resources 14-1
- 14-2. Processing with DD LABEL Subparameter IN or OUT 14-3
- 15-1. Allocation Task for Requesting Data Set Resources 15-1
- 15-2. Affect of Device Status on Allocation 15-2
- 15-3. Unit and Volume Affinity 15-7
- 15-4. Types of JES3 Setup 15-14
- 16-1. Processing Control Task for Requesting Data Set Resources 16-1
- 17-1. End Processing Task for Requesting Data Set Resources 17-1
- 18-1. Identification Task for Requesting Sysout Data Set Resources 18-1
- 19-1. Description Task for Requesting Sysout Data Set Resources 19-1
- 20-1. Performance Control Task for Requesting Sysout Data Set Resources 20-1
- 21-1. Processing Control Task for Requesting Sysout Data Set Resources 21-1
- 22-1. End Processing Task for Requesting Sysout Data Set Resources 22-1
- 23-1. Destination Control Task for Requesting Sysout Data Set Resources 23-1
- 24-1. Output Formatting Task for Requesting Sysout Data Set Resources 24-1
- 25-1. Output Limiting Task for Requesting Sysout Data Set Resources 25-1
- A-1. Area Arrangement of ISAM Data Sets A-5
- A-2. DD Parameters for Retrieving or Extending an ISAM Data Set A-6
- C-1. DD Parameters to Use when Processing VSAM Data Sets C-2
- C-2. DD Parameters to Avoid when Processing VSAM Data Sets C-3



## **Part 1. Introduction**

For your program to execute on the computer and perform the work you designed it to do, your program must be processed by your operating system. Your operating system consists of a base control program (BCP) and the job entry subsystem (JES2 or JES3) installed with it.

For the operating system to process a program, programmers must perform certain job control tasks. These tasks are performed through the job control statements, which are introduced in the first chapter. The job control tasks are introduced in the second chapter. The charts in the third chapter divide these tasks into detailed subtasks. The tasks are:

- Entering jobs
- Processing jobs
- Requesting resources

### **Part 1 Contents**

<b>Chapter 1. Job Control Statements</b>	<b>1-1</b>
<b>Chapter 2. Job Control</b>	<b>2-1</b>
Entering Jobs	2-1
Job Steps	2-1
Jobs	2-1
Input Streams	2-2
Cataloged and In-Stream Procedures	2-3
Steps in a Job	2-4
Jobs with JES2 or JES3 Control Statements	2-4
Processing Jobs	2-4
Requesting Resources	2-4
Data Set Resources	2-4
Sysout Data Set Resources	2-4
<b>Chapter 3. Job Control Tasks</b>	<b>3-1</b>
Task Charts	3-1



## Chapter 1. Job Control Statements

This chapter lists in Figure 1-1 all the job control statements and gives the purpose of each statement.

Statement	Name	Purpose
<b>JCL Statements</b>		
// command	JCL 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.
// CNTL	control	Marks the beginning of one or more program control statements.
// DD	data definition	Identifies and describes a data set.
/*	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.
// ENDCNTL	end control	Marks the end of one or more program control statements.
// 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.
// JOB	job	Marks the beginning of a job; assigns a name to the job.
//	null	Marks the end of a job.
// OUTPUT	output JCL	Specifies the processing options that the job entry subsystem is to use for printing a sysout data set.
// 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.

**Figure 1-1 (Part 1 of 2). Job Control Statements**

# Statements

Statement	Purpose
<b>JES2 Control Statements</b>	
/*\$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 processing options for sysout data set(s).
/*PRIORITY	Assigns a job queue selection priority.
/*ROUTE	Specifies the output destination or the execution node for the job.
/*SETUP	Requests mounting of volumes needed for 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.
<b>JES3 Control Statements</b>	
/**command	Enters JES3 operator commands, except *DUMP and *RETURN, through the input stream.
/**DATASET	Begins an input data set in the input stream.
/**ENDDATASET	Ends the input data set that began with a /**DATASET statement.
/**ENDPROCESS	Ends a series of /**PROCESS statements.
/**FORMAT	Specifies the processing options for a sysout or JES3-managed print or punch data set.
/**MAIN	Defines selected processing parameters for a job.
/**NET	Identifies relationships between predecessor and successor jobs in a dependent job control net.
/**NETACCT	Specifies an account number for a network job.
/**OPERATOR	Sends messages to the operator.
/**PAUSE	Halts the input reader.
/**PROCESS	Identifies a nonstandard job.
/**ROUTE	Specifies the execution node for the job.
/*SIGNOFF	Ends a remote job stream processing session.
/*SIGNON	Begins a remote job stream processing session.

Figure 1-1 (Part 2 of 2). Job Control Statements



## Chapter 2. Job Control

### Entering Jobs

**Job Steps:** You enter a program into the operating system as a **job step**. A job step consists of the job control statements that request and control execution of a program and request the resources needed to run the program. A job step is identified by an EXEC statement. The job step can also contain data needed by the program. The operating system distinguishes job control statements from data by the contents of the records.

**Jobs:** A **job** is a collection of related job steps. A job is identified by a JOB statement.

<b>Job with One Step</b>				
	//JOB1	JOB	ACCT28, 'MAE BIRD'	Identifies job
Step	//STEP1	EXEC	PGM=A	Identifies step, executes program
1	//DD1	DD	*	Defines in-stream data set
			.	
			(data)	
			.	
	//DD2	DD	SYSOUT=H	Defines output data set
<b>Job with Three Steps</b>				
	//JOB1	JOB	ACCT32, 'NICK TULVE'	Identifies job
Step	//STEP1	EXEC	PGM=RDR	Identifies first step, executes program RDR
1	//DDIN1	DD	*	Defines in-stream data set
			.	
			(data)	
			.	
	//DDWRK	DD	DSNAME=A.B.C.,	Requests cataloged data set to be updated
	//		DISP=(MOD,PASS)	
Step	//STEP2	EXEC	PGM=WRTR	Identifies second step, executes program WRTR
2	//DDIN2	DD	DSNAME=*.STEP1.DDWRK,	Requests data set updated in STEP1
	//		DISP=(OLD,PASS)	
	//DDOUT	DD	SYSOUT=D	Defines sysout data set
Step	//STEP3	EXEC	PGM=REPT	Identifies third step, executes program REPT
3	//DDATA	DD	DSNAME=*.STEP2.DDIN2,	Requests data set read in STEP2
	//		DISP=OLD	
	//DDREP	DD	SYSOUT=C	Defines sysout data set

Figure 2-1. Jobs and Job Steps

## Job Control

**Input Streams:** Jobs placed in a series and entered through one input device form an **input stream**. The operating system reads an input stream into the computer from 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. An internal reader is a buffer that is read from a program into the system as through it were an input stream.

```
Input Stream
Job 1 //JOB1 JOB AT45, 'GARY HILL' First job
      //STEP1 EXEC PGM=A33
      //DDA DD DSNAME=CATDS, DISP=OLD
      //DDB DD SYSOUT=A
Job 2 //JOB2 JOB AT87, 'JAN BUSKIRK' Second job
      //STEPA EXEC PGM=REP
      //DD1 DD *
      .
      (data)
      .
Job 3 //DD2 DD SYSOUT=C
      //JOB3 JOB 1726, 'JOYCE GRIFFIN' Third job
      //ST1 EXEC PGM=ADDER
      //DDIN DD DATA
      .
      (data)
      .
      /*
      //DDOUT DD SYSOUT=A
```

Figure 2-2. Job Boundaries in the Input Stream

**Cataloged and In-Stream Procedures:** You often use the same set of job control statements repeatedly with little or no change, for example, to compile, assemble, link-edit, and execute a program. To save time and prevent errors, you can prepare sets of job control statements and place, or catalog, them in a partitioned data set known as a procedure library. Such a set of job control statements in the system procedure library, SYS1.PROCLIB, is called a **cataloged procedure**.

To test a procedure before placing it in the catalog, place it in an input stream and execute it; a procedure in an input stream is called an **in-stream procedure**. The maximum number of in-stream procedures you can code in any job is 15.

<b>In-Stream Procedure</b>		
//JOB1 JOB CT1492,'DAVE HANS'		Starts job
//PTEST PROC		Starts in-stream procedure
//PSTA EXEC PGM=CALC		Identifies first step in procedure
//DDA DD DSNAME=D.E.F,DISP=OLD		Request 3 data sets for first procedure step
//DDB DD DSNAME=DATA1,		
// DISP=(MOD,PASS)		
//DDOUT DD SYSOUT=*		
//PSTB EXEC PGM=PRNT		Identifies second step in procedure
//DDC DD DSNAME=*.PSTA.DDB,		Request 2 data sets for second procedure step
// DISP=OLD		
//DDREP DD SYSOUT=A		
// PEND		Ends in-stream procedure
//STEP1 EXEC PROC=PTEST		First step in JOB1, executes procedure
//PSTA.IN DD *		Adds in-stream data set to procedure step
	PSTA	
	.(data)	
	.	
/*		
<b>Cataloged Procedure: Member MYPROC in SYS1.PROCLIB</b>		
// PROC		Starts cataloged procedure (optional)
//MY1 EXEC PGM=WORK1		Identifies first step in procedure
//MYDDA DD SYSOUT=A		Request 2 data sets for first procedure step
//MYDDB DD SYSOUT=*		
//MY2 EXEC PGM=TEXT5		Identifies second step in procedure
//MYDDC DD DSNAME=F.G.H,DISP=OLD		Request 2 data sets for second procedure step
//MYDDE DD SYSOUT=*		
<b>In-Stream Job that Executes Cataloged Procedure</b>		
//JOB2 JOB , 'BETH MORRISON'		Starts job
//STEP1 EXEC PROC=MYPROC		First step in JOB2, executes procedure
//MY2.MYDDC DD DISP=(OLD,DELETE)		Modifies DD statement MYDDC in procedure step MY2

Figure 2-3. In-Stream and Cataloged Procedures

## Job Control

**Steps in a Job:** A job can be simple or complex; it can consist of one step or of many steps that call many in-stream and cataloged procedures. A job can consist of up to 255 job steps, including all steps in any procedures that the job calls. Specification of a greater number of steps produces unpredictable results.

**Jobs with JES2 or JES3 Control Statements:** The JES2 or JES3 control statements are placed in in-stream jobs. These statements cannot appear in cataloged or in-stream procedures.

### *Input Stream Job with JES2 Statements*

```
/*PRIORITY 9                                JES2 statement
//JOBH JOB AT45,'MIKE COLLINS'
/*JOBPARM BYTES=100,COPIES=5                JES2 statement
//STEP1 EXEC PGM=REPORT1
//DDA DD DSNAME=DATA2,DISP=OLD
//DDB DD SYSOUT=A
```

### *Input Stream Job with JES3 Statements*

```
//JOBH JOB AT45,'MIKE COLLINS'
//*MAIN BYTES=100                            JES3 statements
//*FORMAT PR,DDNAME=,COPIES=5
//STEP1 EXEC PGM=REPORT1
//DDA DD DSNAME=DATA2,DISP=OLD
//DDB DD SYSOUT=A
```

Figure 2-4. JES Control Statements in Jobs

## Processing Jobs

The operating system performs many job control tasks automatically. You can influence the way your job is processed by the JCL and JES2 or JES3 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.

## Requesting Resources

**Data Set Resources:** To execute a program, you must request the data sets needed to supply data to the program and to receive output records from the program.

**Sysout Data Set Resources:** A sysout data set is a system-handled output data set. This data set is placed temporarily on direct access storage. Later, at the convenience of the system, the system prints it, punches it, or sends it to a specified location. Because sysout data sets are processed by the system, the programmer can specify many parameters to control that processing.

## Chapter 3. Job Control Tasks

### Task Charts

The following charts list the job control tasks in four groups:

- Entering jobs in Figure 3-1 on page 3-2
- Processing jobs in Figure 3-2 on page 3-5
- Requesting data set resources in Figure 3-3 on page 3-6
- Requesting sysout data set resources in Figure 3-4 on page 3-9

For each task, the charts list the parameters and statements that can be used to perform it. In many cases, the same task can be performed using different parameters on different statements. Where a parameter can appear on both a JOB and EXEC statement, it applies to the entire job when coded on the JOB statement but only to a step when coded on an EXEC statement.

The system is designed to enable users to perform many types of job control in many ways. To allow this flexibility, only two job entry tasks are required:

- **Identification:** The job must be identified in the *jobname field* of a JOB statement.
- **Execution:** The program or procedure to be executed must be named in a PGM or PROC parameter on an EXEC statement.

Therefore, the following statements are the minimum needed to perform a job control task:

```
//jobname JOB
//          EXEC {PGM=program-name  }
                {PROC=procedure-name}
```

# Tasks

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
<b>Identification</b>					
of job	jobname field		null statement (JES3 only)		
of step		stepname field			
of procedure			PROC PEND		
of account	accounting information or pano in JOB JES2 accounting information	ACCT		/*NETACCT	//*NETACCT
of programmer	programmer's name and room in JOB JES2 accounting information USER			ROOM on /*JOBPARM	PNAME, BLDG, DEPT, ROOM, and USERID on /*NETACCT
<b>Execution</b>					
of program		PGM			
of procedure		PROC			
when restarting and with checkpointing	RESTART RD	RD	SYSCHK DD	RESTART on /*JOBPARM	FAILURE and JOURNAL on /*MAIN
deadline or periodic					DEADLINE on /*MAIN
when dependent on other jobs					/*NET
at remote node				/*ROUTE XEQ /*XEQ /*XMIT	/*ROUTE XEQ

Figure 3-1 (Part 1 of 3). Tasks for Entering Jobs

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
<b>Job input control</b>					
by holding job entrance	TYPRUN CLASS				HOLD, UPDATE, or CLASS on <b>//*MAIN</b> <b>//*NET</b>
by holding local input reader					<b>//*PAUSE</b>
by copying input stream (JES2 only)	TYPRUN CLASS				
from remote work station				<b>/*SIGNON</b> <b>/*SIGNOFF</b>	<b>/*SIGNON</b> <b>/*SIGNOFF</b>
<b>Communication</b>					
from JCL to system			Command	<b>/*\$command</b>	<b>/**command</b>
from JCL to operator				<b>/*MESSAGE</b>	<b>/*OPERATOR</b>
from JCL to programmer	Comment field unless no parameter field	Comment field	<b>/**comment</b> , also comment field on all statements but null		Comment field on <b>/*ENDPROCESS</b> and <b>/*PAUSE</b>
from JCL to program		PARM			
from system to operator					FETCH on <b>/*MAIN</b> WARNING on BYTES, CARDS, LINES, and PAGES on <b>/*MAIN</b>
from system to TSO userid	NOTIFY			<b>/*NOTIFY</b>	ACMAIN on <b>/*MAIN</b> with JOB NOTIFY
from TSO userid to system					USER on <b>/*MAIN</b>
from functional subsystem to programmer			PIMSG on OUTPUT JCL		
through job log	MSGCLASS MSGLEVEL log in JOB JES2 accounting information		JESDS on OUTPUT JCL	NOLOG on <b>/*JOBPARM</b>	

Figure 3-1 (Part 2 of 3). Tasks for Entering Jobs

# Tasks

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
<b>Protection</b>					
through RACF	GROUP PASSWORD USER				
<b>Resource control</b>					
of program library			JOBLIB DD STEPLIB DD DD defining PDS member		
of procedure library				PROCLIB on /*JOBPARM	PROC and UPDATE on /*MAIN
of address space	REGION ADDRSPC	REGION ADDRSPC			LREGION on /*MAIN
of processor				SYSAFF on /*JOBPARM	SYSTEM on /*MAIN
of spool partition					SPART and TRKGRPS on /*MAIN

Figure 3-1 (Part 3 of 3). Tasks for Entering Jobs



TASKS FOR PROCESSING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
<b>Processing control</b>					
by terminating execution	COND	COND			CANCEL in BYTES, CARDS, LINES, and PAGES on /*MAIN
by timing execution	TIME or time in JOB JES2 accounting information	TIME		TIME on /*JOBPARM	
for testing: (1) by altering usual processing  (2) by dumping after error	TYPRUN CLASS	PGM = IEFBR14  PGM = JCLTEST PGM = JSTTEST (JES3 only)	SYSABEND DD SYSMDUMP DD SYSUDUMP DD  To format dump on 3800 Printing Subsystem, FCB = STD3 and CHARS = DUMP on dump DD		/*PROCESS /*ENDPROCESS  DUMP in BYTES, CARDS, LINES, and PAGES on /*MAIN
<b>Performance control</b>					
by job class assignment	CLASS				CLASS on /*MAIN
by selection priority	PRTY			/*PRIORITY	
by dispatching priority		DPRTY			
by performance group assignment	PERFORM				
by I/O-to-processing ratio					IORATE on /*MAIN

Figure 3-2. Tasks for Processing Jobs

# Tasks

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Identification</b>					
of data set	DSNAME				UPDATE on /*MAIN
of in-stream data set	* or DATA SYSIN DD DLM		/* or xx delimiter		/*DATASET /*ENDDATASET
of data set on 3540 Diskette Input/Out- put Unit	DSID				
through catalog	JOBCAT DD STEPDAT DD				
through label	label-type on LABEL				
by location on tape	data-set- sequence-number on LABEL				
as TCAM message data set	QNAME				
from or to terminal	TERM				
<b>Description</b>					
of status	DISP				
of data attributes	DCB AMP				

Figure 3-3 (Part 1 of 3). Tasks for Requesting Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Protection</b>					
through RACF	PROTECT				
for ISO/ANSI/FIPS Version 3 tapes	ACCODE				
by passwords	PASSWORD and NOPWREAD on LABEL				
of access to BSAM and BDAM data sets	IN and OUT on LABEL				
<b>Allocation</b>					
of device	UNIT		CLASS on JOB (JES3 only)		SETUP, MSS, and CLASS on /*MAIN
of tape or direct access volume	VOLUME MSVGP				EXPDTCHK and RINGCHK on /*MAIN
of direct access space	SPACE				
of virtual I/O	UNIT DSNAME = temporary data set				
with deferred volume mounting	DEFER on UNIT				
with volume pre-mounting				/*SETUP	
dynamic			DYNAMNBR on EXEC		

Figure 3-3 (Part 2 of 3). Tasks for Requesting Data Set Resources

# Tasks

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Processing control</b>					
by suppressing processing	DUMMY NULLFILE on DSNAME				
by postponing specification	DDNAME				
with checkpointing	CHKPT SYSCKEOV DD				
by subsystem	SUBSYS CNTL		CNTL ENDCNTL		
by TCAM job or task	QNAME				
<b>End processing</b>					
deallocation	FREE				
disposition of data set	DISP RETPD and EXPDT on LABEL				
release of unused direct access space	RLSE on SPACE				
disposition of volume	RETAIN and PRIVATE on VOLUME				

Figure 3-3 (Part 3 of 3). Tasks for Requesting Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Identification</b>					
as a sysout data set	SYSOUT				
of output class	class on SYSOUT	CLASS	MSGCLASS on JOB with SYSOUT = * or CLASS = * and SYSOUT = (,)		
of data set on 3540 Diskette Input/Output Unit	DSID				
<b>Description</b>					
of data attributes	DCB				
<b>Performance control</b>					
by queue selection		PRTY			

Figure 3-4 (Part 1 of 4). Tasks for Requesting Sysout Data Set Resources

# Tasks

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Processing control</b>					
with additional parameters	OUTPUT code-name on SYSOUT	DEFAULT			
with other data sets	class on SYSOUT	THRESHLD (JES3 only) GROUPID (JES2 only)			
by external writer	writer-name on SYSOUT	WRITER			
by mode		PRMODE			
by holding	HOLD class on SYSOUT	CLASS			
by suppressing output	DUMMY class on SYSOUT				
with checkpointing		CKPTLINE CKPTPAGE CKPTSEC			
by Print Services Facility (PSF)		FORMDEF PAGEDEF			

Figure 3-4 (Part 2 of 4). Tasks for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
End processing					
deallocation	FREE				
Destination control					
to local or remote device or to another node	DEST class on SYSOUT	DEST COMPACT		/*ROUTE PRINT /*ROUTE PUNCH	ORG on /*MAIN
to another processor					ACMAIN on /*MAIN
to internal reader	INTRDR as writer-name on SYSOUT		/*EOF /*DEL /*PURGE /*SCAN		
to terminal	TERM				

Figure 3-4 (Part 3 of 4). Tasks for Requesting Sysout Data Set Resources

# Tasks

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Output formatting</b>					
to any printer	COPIES FCB form-name on SYSOUT  UCS	COPIES FCB FORMS LINECT (JES2 only) UCS CONTROL	forms, copies, and linect on JOB JES2 accounting information	COPIES, FORMS, and LINECT on /*JOBPARM	
to 3800 Printing Sub- system, in addition to most of printer parameters	BURST CHARS FLASH MODIFY DCB=OPTCD=J	BURST CHARS FLASH MODIFY TRC		BURST on /*JOBPARM	
to 3211 Printer with indexing feature		INDEX (JES2 LINDEX only)			
to punch	COPIES FCB form-name on SYSOUT DCB=FUNC=I	COPIES FCB FORMS			
of dumps on 3800 Printing Subsys- tem	CHARS=DUMP FCB=STD3	CHARS=DUMP FCB=STD3			
<b>Output limiting</b>	OUTLIM		lines and cards on JOB JES2 accounting information	BYTES, CARDS, LINES, and PAGES on /*JOBPARM	BYTES, CARDS, LINES, and PAGES on /*MAIN

Figure 3-4 (Part 4 of 4). Tasks for Requesting Sysout Data Set Resources



## **Part 2. Tasks for Entering Jobs**

This part describes how to enter jobs into the system. The tasks required to enter a job are:

- Identification
- Execution.

Other tasks can optionally be performed:

- Job input control
- Communication
- Protection
- Resource control

### **Part 2 Contents**

<b>Chapter 4. Identification</b>	4-1
Identification of Job	4-2
Examples	4-2
Identification of Step	4-2
Examples	4-2
Identification of Procedure	4-2
Examples	4-3
Identification of Account	4-3
For Local Execution	4-3
Examples	4-3
For Remote Execution	4-4
Examples	4-4
Identification of Programmer	4-4
Examples	4-4
<b>Chapter 5. Execution</b>	5-1
Execution of Program	5-1
Examples	5-2
Execution of Procedure	5-2

## Part 2

Examples	5-2
Execution when Restarting and with Checkpointing	5-2
Restarting after Abnormal Termination	5-2
Use of Restart	5-3
Examples	5-3
Restarting When the System Failed in a JES2 System	5-3
Examples	5-3
Restarting When the System Failed in a JES3 System	5-3
Examples	5-4
Deadline or Periodic Execution in a JES3 System	5-4
Use of Deadline Scheduling	5-4
Examples	5-4
Use of Periodic Scheduling	5-4
Examples	5-4
Execution when Dependent on Other Jobs in a JES3 System	5-5
External Dependencies	5-5
Testing a Net	5-5
Examples	5-6
Execution at Remote Node	5-7
Considerations when Submitting a Remote Job	5-8
Examples	5-8
<b>Chapter 6. Job Input Control</b>	6-1
Job Input Control by Holding Job Entrance	6-1
Use of Job Holding	6-2
Examples	6-2
Job Input Control by Holding Local Input Reader in a JES3 System	6-2
Example	6-2
Job Input Control by Copying Input Stream in a JES2 System	6-3
Examples	6-3
Job Input Control from Remote Work Station	6-3
JES2 Remote Job Entry	6-3
Remote Job Entry Stations	6-4
JES3 Remote Job Processing	6-4
Remote Work Stations	6-4
<b>Chapter 7. Communication</b>	7-1
Communication from JCL to System	7-2
Examples	7-2
Communication from JCL to Operator	7-2
Examples	7-2
Communication from JCL to Programmer	7-2
Examples	7-3
Communication from JCL to Program	7-3
Examples	7-3
PARM Values for IBM-Supplied Programs	7-3
Communication from System to Operator	7-3
Messages during Volume Mounting	7-3
Examples	7-4
Messages when Job Exceeds Output Limit	7-4
Use of Warning Messages	7-4
Examples	7-4
Communication from System to Time Sharing Userid	7-4
Examples	7-5

Communication from Time Sharing Userid to a JES3 System	7-5
Examples	7-5
Communication from Functional Subsystem to Programmer	7-6
Example	7-6
Communication through Job Log	7-6
Examples	7-7
Printing Job Log and Sysout Data Sets Together	7-7
Examples	7-8
<b>Chapter 8. Protection</b>	8-1
Protection through RACF	8-1
Examples	8-1
<b>Chapter 9. Resource Control</b>	9-1
Resource Control of Program Library	9-1
System Library	9-2
Private Library	9-2
Use of Private Libraries	9-2
Creating a Private Library	9-2
Adding Members to a Private Library	9-2
Example of Creating and Adding to a Private Library	9-3
Retrieving an Existing Private Library	9-3
Example of Retrieving Job and Step Libraries	9-3
Concatenating Private Libraries	9-4
Example of Concatenated Libraries	9-4
Temporary Library	9-4
Creating a Temporary Library	9-4
Example	9-4
Resource Control of Procedure Library	9-5
Updating Procedure Library	9-5
Examples	9-5
Resource Control of Address Space	9-6
Types of Storage	9-6
Virtual Storage	9-6
Real Storage	9-6
Requesting Amount and Type of Storage	9-6
Region Size for Virtual Storage	9-7
Region Size for Real Storage	9-7
Examples	9-8
Requesting Amount of Logical Storage in a JES3 System	9-8
Example	9-8
Resource Control of the Processor	9-8
Selecting a Processor in JES2	9-8
Independent Mode	9-9
Examples	9-9
Selecting a Processor in JES3	9-9
Relationship to Other Parameters	9-10
Examples	9-10
Resource Control of Spool Partitions in a JES3 System	9-10
Examples	9-11



## Chapter 4. Identification

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
<b>Identification</b>					
of job	jobname field		null statement (JES3 only)		
of step		stepname field			
of procedure			PROC PEND		
of account	accounting information or pano in JOB JES2 accounting information	ACCT		/*NETACCT	/*NETACCT
of programmer	programmer's- name and room in JOB JES2 accounting information USER			ROOM on /*JOBPARM	PNAME, BLDG, DEPT, ROOM, and USERID on /*NETACCT

Figure 4-1. Identification Task for Entering Jobs

## Identification

### Identification of Job

Each job must be identified in the jobname field of the JOB statement. This identification is required and is coded:

```
//jobname JOB
```

The next JOB statement or the end of the input stream identifies the end of a job. A null statement can identify the end of a job or input stream.

#### *Examples*

---

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

This fifth statement is a null statement.

---

### Identification of Step

A step name is required on only certain EXEC statements. In practice, name all steps. The system uses the step name in messages. If you omit the step name, the system leaves this field blank in messages, making it difficult to decide what step caused each message. A step name is coded:

```
//stepname EXEC
```

#### *Examples*

---

```
//STEP1    EXEC  PGM=A  
//CHECK    EXEC  PROC=MHB15  
//A$9      EXEC  PGM=RPTWRT  
//MYPROGRM EXEC  PGM=CALC
```

---

### Identification of Procedure

For an in-stream procedure, identify the beginning with a PROC statement and the end with a PEND statement. Code a name on the PROC statement.

For a cataloged procedure, a PROC statement is optional and a PEND statement is invalid. A PROC statement does not identify a cataloged procedure; the procedure is called by its member name or alias in the procedure library. However, use the PROC statement to assign default values for all symbolic parameters in the procedure. Then, if the calling EXEC statement fails to assign a value to all symbolic parameters, the step will not fail.

## *Examples*

---

For in-stream procedures:

```
//PAYROLL PROC
      :
      :
//      PEND
//DESK3  PROC  A=NEWYORK,F=3350,C=(OLD,CATLG,DELETE)
      :
      :
//ENDING  PEND  THIS STATEMENT ENDS IN-STREAM PROCEDURE DESK3.
```

For cataloged procedures:

```
//      PROC  UT=3800,FM=J287,DT=LOCAL
```

---

## Identification of Account

### For Local Execution

In JES initialization parameters, the installation specifies whether or not accounting information is required in the accounting information parameter on the JOB statement and/or the ACCT parameter on the EXEC statement. The installation decides what accounting information is needed and the format for the information.

### *Examples*

---

```
//J28  JOB (12A75,DEPTD58,921)
//XYZ  JOB '12A75,DEPTD58,921'
```

If a subparameter contains special characters:

```
//GHI  JOB (12A75,'DEPT/D58',921)
//JKL  JOB '12A75,DEPT/D58,921'
```

If only an account number is coded:

```
//MNO  JOB 12A75
//PQR  JOB '12A.75'
```

If the account number is omitted:

```
//STU  JOB (,DEPTD58,921)
```

---

# Identification

## For Remote Execution

The JES2 /\*NETACCT statement and the JES3 /\*NETACCT statement supply accounting information for jobs sent to remote nodes for execution.

### *Examples*

---

For remote execution in a JES2 system:

```
/*NETACCT 27FD16
```

For remote execution in a JES3 system:

```
/*NETACCT PNAME=FKRUPA,ACCT=27FD16,BLDG=921,DEPT=D58,  
/*NETACCT ROOM=2T13,USERID=DDFKPGMR
```

---

## Identification of Programmer

In JES initialization parameters, the installation specifies if a programmer's-name parameter is required on the JOB statement. The installation decides what the parameter must contain.

The USER parameter can be coded on the JOB statement to identify the person submitting the job. Normally, this parameter is used by the Resource Access Control Facility (RACF); however, it is also used by other system components, including the system resources manager (SRM).

### *Examples*

---

```
//ABC JOB ,L.GORDON  
//DEF JOB , 'L GORDON'  
//GHI JOB , 'SP/4 L. GORDON'  
//JKL JOB , 'DEPT. 7202'  
//MNO JOB ACCT15, 'DON PIZZUTO', USER=ID32DBP
```

---



## Chapter 5. Execution

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
<b>Execution</b>					
of program		PGM			
of procedure		PROC			
when restarting and with checkpointing	RESTART RD	RD	SYSCHK DD	RESTART on /*JOBPARM	FAILURE and JOURNAL on /*MAIN
deadline or periodic					DEADLINE on /*MAIN
when dependent on other jobs					/*NET
at remote node				/*ROUTE XEQ /*XEQ /*XMIT	/*ROUTE XEQ

Figure 5-1. Execution Task for Entering Jobs

### Execution of Program

All programs to be executed must reside in a library, which is a partitioned data set. The installation should maintain a list of programs available in its libraries. Libraries are of three types:

- System libraries: SYS1.LINKLIB
- Private libraries, specified in a JOBLIB or STEPLIB DD statement.
- Temporary libraries, created in a previous step of the job.

For information about libraries, see “Resource Control of Program Library” on page 9-1.

## Execution

Execute a program in a system or private library by coding:

```
//stepname EXEC PGM=program-name
```

Execute a program in a temporary library by coding:

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

### Examples

---

```
//ST1 EXEC PGM=MYPROG  
//DSPROG DD DSN=DSNAME=PDS1(MEMP)  
//ST2 EXEC PGM=*.ST1.DSPROG
```

---

## Execution of Procedure

A procedure to be executed must be:

- In-stream procedure, located in the input stream before the EXEC statement that calls it.
- Cataloged procedure, located in the system catalog: SYS1.PROCLIB.

Execute an in-stream or cataloged procedure by coding:

```
//stepname EXEC PROC=procedure-name  
//stepname EXEC procedure-name
```

### Examples

---

```
//ST1 EXEC PROC=PROCA  
//STEP9 EXEC PROC=DAILY
```

---

## Execution when Restarting and with Checkpointing

### Restarting after Abnormal Termination

If a job terminates abnormally, the checkpoint/restart facilities allow you to restart the job, as follows:

- Automatic step restart, that is, restart by the system from the beginning of a job step.
- Automatic checkpoint restart, that is, restart by the system from a checkpoint within a job step.
- Deferred step restart, that is, restart at a later time from the beginning of a job step.
- Deferred checkpoint restart, that is, restart at a later time from a checkpoint within a job step.

Restarts are controlled by:

- RD parameters on JOB and EXEC statements.
- Checkpoints, if written. Each time an assembler CHKPT macro is executed, a checkpoint is written.
- The job journal, which is required for a restart. In a JES3 system, the programmer can code a JOURNAL parameter on the JES3 `//*MAIN` statement to control whether JES3 creates a journal for the job.
- In deferred restarts, a RESTART parameter on the JOB statement for the restarting job and a SYSCHK DD statement to identify the data set containing the checkpoint written in response to an assembler CHKPT macro.

**Use of Restart:** Either form of restart saves having to execute the job from its beginning. If the job is long, restarting can save a lot of time and computer resources.

### Examples

---

```
//J1    JOB    , 'B. MORRISON' ,RD=RNC
//J2    JOB    , 'H. MORRILL'
//S1    EXEC   PGM=TESTING ,RD=R
//S2    EXEC   PGM=TESTED ,RD=NC
```

---

### Restarting When the System Failed in a JES2 System

If (1) the job was executing when the system failed and the operator re-IPLs the system with a JES2 warm start and (2) the job cannot restart from a step or a checkpoint, JES2 requeues the job for execution if `RESTART=Y` is in the JES2 `/*JOBPARM` statement. Re-execution is from the beginning of the job.

### Examples

---

```
//J3    JOB    , 'J. BUSKIRK'
/*JOBPARM  RESTART=Y
      :
```

---

### Restarting When the System Failed in a JES3 System

If the job was executing when the system failed, the FAILURE parameter on the JES3 `//*MAIN` statement tells JES3 how to handle the job. The job can be restarted, cancelled, held, or printed and then held for restart.

## Execution

### *Examples*

---

```
//J4 JOB , 'G. HILL', RD=NC  
//*MAIN FAILURE=RESTART  
:
```

---

## Deadline or Periodic Execution in a JES3 System

Use the DEADLINE parameter on the JES3 `//*MAIN` statement to execute your job by a certain time or periodically every week, month, or year. As the deadline approaches, JES3 increases the job's priority until it is executed. The priority is increased according to the installation-defined algorithm requested in the second subparameter.

### Use of Deadline Scheduling

The purpose of deadline scheduling is to help JES3 use available resources best. For example, if you work first shift and submit a job at the end of the day, you do not need output until the next morning. Specify 7 a.m. of the next day in the DEADLINE parameter and assign the job a low priority. JES3 can schedule the job any time during the night when the resources are available. But, if the job has not been scheduled by several hours before 7 a.m., JES3 increases its priority. JES3 will increase the job's priority periodically until it is selected for execution by 7 a.m.

### *Examples*

---

To execute a job by 7 a.m. on January 20, 1986, code:

```
//*MAIN DEADLINE=(0700,B,012086)
```

---

### Use of Periodic Scheduling

The purpose of periodic scheduling is to run certain weekly, monthly, or yearly programs automatically.

### *Examples*

---

To execute a job by 2 p.m. every Friday, code:

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

---

## Execution when Dependent on Other Jobs in a JES3 System

Use dependent job control (DJC) when jobs must be executed in a specific order. 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, code a JES3 `/**NET` statement in each job. Jobs in a net are of two types:

- Predecessor jobs, which must be completed before another job.
- Successor jobs, which must not be executed until one or more jobs are completed.

Using parameters on the `/**NET` statement, you can make execution of a job depend on how a predecessor terminated: normally or abnormally. When a predecessor job completes, a successor job:

- Can have the count of predecessor jobs it is waiting for decreased by one. When the count reaches zero, the successor job is queued for execution.
- Can be flushed from the system. The successor job and all of its successors are canceled, printed, and flushed from the system.
- Can be retained until the operator releases it. The successor job and all of its successors are kept from being scheduled. The job is released only when its immediate predecessor is resubmitted or the operator decreases the predecessor job number.

**External Dependencies:** If your job depends on external events, you can specify a count of predecessor jobs that is one greater than needed. The system will hold the job because the count cannot reach zero. When the external event occurs, the operator can issue a `*MODIFY,N` command to reduce the number so that the job will execute.

**Testing a Net:** To test a net without executing the programs, substitute the following for each actual EXEC statement:

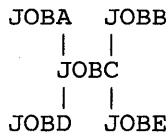
```
/**TESTx EXEC PGM=IEFBR14
```

# Execution

## Examples

---

To set up a dependent job net, first draw a diagram of the dependencies:



Give the net a name: XMP1. This is the `//*NET` statement NETID parameter.

Then list each job and its predecessors and successors:

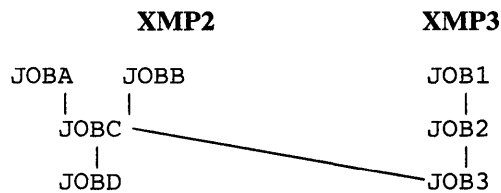
jobname	Predecessors //*NET NHOLD	Successors //*NET RELEASE
JOBA	0	JOBC
JOBB	0	JOBC
JOBC	2	JOBD, JOBE
JOBD	1	none
JOBE	1	none

Finally, code a `//*NET` statement to appear in each job:

```
//JOBA JOB ...
//*NET NETID=XMP1,RELEASE=(JOBC)
//S1 EXEC ...
.
.
//JOBB JOB ...
//*NET NETID=XMP1,RELEASE=(JOBC)
//SA EXEC ...
.
.
//JOBC JOB ...
//*NET NETID=XMP1,NHOLD=2,RELEASE=(JOBD,JOBE)
//S1 EXEC ...
.
.
//JOBD JOB ...
//*NET NETID=XMP1,NHOLD=1
//SA EXEC ...
.
.
//JOBE JOB ...
//*NET NETID=XMP1,NHOLD=1
//S1 EXEC ...
.
```

---

This example shows two nets. JOB3 in net XMP3 depends on JOBC in net XMP2.



jobname	Predecessors //*NET NHOLD	Successors //*NET RELEASE
JOBA	0	JOBC
JOBB	0	JOBC
JOBC	2	JOB3
JOBD	1	none
JOB1	0	JOB2
JOB2	1	JOB3
JOB3	2	none

The `//*NET` statements for each job are:

```

For JOBA:  /**NET NETID=XMP2,RELEASE=(JOBC)
For JOBB:  /**NET NETID=XMP2,RELEASE=(JOBC)
For JOBC:  /**NET NETID=XMP2,NHOLD=2,NETREL=(XMP3,JOB3)
For JOBD:  /**NET NETID=XMP2,NHOLD=1
For JOB1:  /**NET NETID=XMP3,RELEASE=(JOB2)
For JOB2:  /**NET NETID=XMP3,NHOLD=1,RELEASE=(JOB3)
For JOB3:  /**NET NETID=XMP3,NHOLD=2
  
```

## Execution at Remote Node

You can enter a job through your system to execute on another system by coding one of the following statements. The job can be entered through an input reader, an internal reader, a TSO terminal, or an RJE (remote job entry) or RJP (remote job processing) terminal or work station.

### When Entered through a JES2 System:

- And received by a JES2 system, code one of the following:

```

/*ROUTE XEQ node
/*XEQ node
  
```

- And received by a JES2 system or a JES3 system, code:

```

/*XMIT node
  
```

## Execution

- And received by a VM system with an MVS system running as a guest, code one of the following:

```
/*ROUTE XEQ node.vmguestid
/*XEQ node.vmguestid
/*XMIT node.vmguestid
```

### When Entered through a JES3 System:

- And received by a system that can process MVS JCL, code:

```
//*ROUTE XEQ node
```

- And received by a VM system with an MVS system running as a guest, code:

```
//*ROUTE XEQ node.vmguestid
```

### Considerations when Submitting a Remote Job

When submitting a job for remote execution, find out the installation-determined attributes of the executing system. Code these values in your JCL for the job.

**The content and format of the JOB statement:** Code on the JOB statements that the executing system will process the executing system's parameters.

**The JES of the executing system:** Code your JES control statements and JCL parameters for the executing system's JES.

**The content of SYS1.PROCLIB in the executing system:** Call only procedures available in the executing system.

**The data sets at the executing system:** Use only data sets that are available at the executing system, with the DD parameters that the executing system requires.

**Installation-specific device names:** Code only UNIT names used by the executing system.

**The sysout classes at the executing system:** Specify the executing system's sysout classes that have the attributes you need.

**The job classes at the executing system:** Specify the executing system's job class that has the attributes you need.

### Examples

---

```
//MYJOB      JOB 27D15,'DON SCHOFER'
/*ROUTE XEQ FARSYS
//THEIRJOB  NJB (DLD1,2E44),'POK LAB'
/*MAIN JOURNAL=YES
//S1       EXEC PROC=RR23,A=3350,
//         C=25,DP=OLD
/*
```

Processed by submitting JES3 location  
Sends the following job to FARSYS  
Sent as JOB statement; processed by FARSYS

---



## Chapter 6. Job Input Control

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
<b>Job input control</b>					
by holding job entrance	TYPRUN CLASS				HOLD, UPDATE, or CLASS on <code>//*MAIN</code> <code>//*NET</code>
by holding local input reader					<code>//*PAUSE</code>
by copying input stream (JES2 only)	TYPRUN CLASS				
from remote work station				<code>/*SIGNON</code> <code>/*SIGNOFF</code>	<code>/*SIGNON</code> <code>/*SIGNOFF</code>

Figure 6-1. Input Control Task for Entering Jobs

### Job Input Control by Holding Job Entrance

If a job must wait for an external event before it can execute, use one of the following. JES holds the job until the system operator releases it or until an event occurs.

#### In a JES2 system

- TYPRUN=HOLD or TYPRUN=JCLHOLD on the JOB statement. The operator must release the job.
- A JOB statement CLASS that requests a job class defined during JES2 initialization as held. The operator must release the job.

#### In a JES3 system

- TYPRUN=HOLD or CLASS on the JOB statement or HOLD=YES or CLASS on the `//*MAIN` statement. The operator must release the job.

## Job Input Control

- A job in a dependent job net; see “Execution when Dependent on Other Jobs in a JES3 System” on page 5-5. JES3 releases the job when the other job(s) complete execution, or the operator releases the job.
- UPDATE on the `//*MAIN` statement of another job, if this job would use the procedure library being updated or any library concatenated to it. JES3 releases the job when the updating job completes execution.

**Use of Job Holding:** You may need to delay execution of a job for several reasons. For example:

- If one job is updating a data set that another job must use.
- If the resources a job requires may not be available until an external event occurs.

**Note:** You cannot depend on job priorities to control the order in which jobs execute. The priority specified in the JOB statement `PRTY` parameter or in the JES2 `/*PRIORITY` statement affects the selection order. It does not guarantee that a job with a higher priority will complete execution before a job with a lower priority is started.

### Examples

---

```
//J1 JOB , 'J. COLE', TYPRUN=HOLD
//J2 JOB ACCT1734, 'T. CURATOLO', CLASS=H

//*MAIN HOLD=YES
//*MAIN UPDATE=DS3
```

---

## Job Input Control by Holding Local Input Reader in a JES3 System

Use a `/*PAUSE` statement to halt an input reader. JES3 issues a message and waits for the operator to issue a `*START` command or for a remote work station with console level 15 to send a start message.

### Example

---

```
//*PAUSE
//FIRST JOB , 'D. SCHOFER'
      :
```

---

## Job Input Control by Copying Input Stream in a JES2 System

Code one of the following on the JOB statement to copy an input job, without executing any steps. While copying the input stream, JES2 scans the JCL for syntax errors.

- TYPRUN=COPY
- A CLASS job class defined during JES2 initialization as containing jobs to be copied without execution.

In both cases, JES2 places the copy of the input stream in a sysout data set. The sysout data set is in the class specified in the JOB statement MSGCLASS parameter. Pick the MSGCLASS class to control how the copied input stream is to be processed, as follows:

- By JES2 or by an external writer.
- Scheduled for immediate output or held because the message class is held. If held, the sysout data set is available to the TSO OUTPUT command.

### *Examples*

---

```
//CPYJ1 JOB 1589D10,'J. PERLMAN',TYPRUN=COPY
//CPYJ2 JOB , 'C. SARDO',CLASS=P
```

---

## Job Input Control from Remote Work Station

### JES2 Remote Job Entry

JES2 remote job entry (RJE) allows a remote work station to submit a job to a distant system and have the job processed by the system's JES2. The output can be retained at the host system, sent to the work station, or sent to another location. JES2 processes a remote job as if it had been submitted locally. The remote station becomes a logical extension of the computer system that processes its jobs.

JES2 supports two ways of communicating with RJE remote stations:

- Through **systems network architecture synchronous data link control (SNA/SDLC)** protocol. SNA stations gain access to JES2 through VTAM.
- Through **binary synchronous communication (BSC)** protocol. Communication between the local processor and a BSC RJE station uses a JES2 facility called *multi-leaving*. Multi-leaving allows 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*.

## Job Input Control

JES2 expects the remote station to be under the control of a remote operator. The RJE stations can consist of two types of devices:

- **Remote terminal**, which does not have a processor. A remote terminal, for example a 2780 or 2770, can be used to enter jobs into and receive data from JES2.
- **Remote work station**, which has a processor. A processor, for example a System/3 or System/370, executes a JES2-generated program that allows the processor to send jobs to and receive data from JES2. The remote work station also includes printers, punches, card readers, and a console.

**Remote Job Entry Stations:** During JES2 initialization, installations can configure remote lines as dedicated or nondedicated. For nondedicated remote lines, use the following to notify JES2 that you wish to begin and end a remote job stream processing session:

- For SNA remote work stations: the LOGON command to begin and either the LOGOFF command or the JES2 /\*SIGNOFF control statement to end.
- For BSC remote work stations: the JES2 /\*SIGNON control statement to begin and the JES2 /\*SIGNOFF control statement to end.

For a discussion of the LOGON and LOGOFF commands, refer to *SPL: JES2 Initialization and Tuning* and *SPL: VTAM*.

### JES3 Remote Job Processing

JES3 remote job processing (RJP) allows a remote work station to submit a job through a data link to a distant global processor and have the job processed by the system's JES3. The output can be retained at the host system, sent to the work station, or sent to another location. JES3 processes a remote job as if it had been submitted locally.

Devices attached to a processor by channels are **local devices**; devices attached to a processor by a data link are **remote devices**.

JES3 supports two ways of communicating with RJP remote devices:

- Through **systems network architecture synchronous data link control (SNA/SDLC)** protocol.
- Through **binary synchronous communications (BSC)** protocol.

**Remote Work Stations:** During JES3 initialization, installations can configure remote lines as dedicated or nondedicated. For nondedicated remote lines, use the following to notify JES3 that you wish to begin and end a remote job stream processing session:

- For SNA remote work stations: the LOGON command to begin and either the LOGOFF command or the JES3 /\*SIGNOFF control statement to end.
- For BSC remote work stations: the JES3 /\*SIGNON control statement to begin and the JES3 /\*SIGNOFF control statement to end.

For a discussion of the LOGON and LOGOFF commands, refer to *SPL: JES3 Initialization and Tuning* and *SPL: VTAM*.

## Chapter 7. Communication

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Communication					
from JCL to system			Command	/*\$command	/**command
from JCL to operator				/*MESSAGE	/**OPERATOR
from JCL to programmer	Comment field unless no parameter field	Comment field	/*comment, also comment field on all statements but null		Comment field on /**ENDPROCESS and /**PAUSE
from JCL to program		PARM			
from system to operator					FETCH on /**MAIN WARNING on BYTES, CARDS, LINES, and PAGES on /**MAIN
from system to TSO userid	NOTIFY			/*NOTIFY	ACMAIN on /**MAIN with JOB NOTIFY
from TSO userid to system					USER on /**MAIN
from functional subsystem to programmer			PIMSG on OUTPUT JCL		
through job log	MSGCLASS MSGLEVEL log in JOB JES2 accounting information		JESDS on OUTPUT JCL	NOLOG on /*JOBPARM	

Figure 7-1. Communication Task for Entering Jobs

## Communication

### Communication from JCL to System

Use the following to communicate from your JCL to the system:

- In a JES2 system, the JCL command statement to enter system operator commands and the JES2 /\*\$command statement to enter JES2 commands.
- In a JES3 system, the JES3 /\*\*command statement to enter JES3 commands.

The system executes any in-stream command as soon as it is read. Therefore, the command will not be synchronized with the execution of any job or step.

#### *Examples*

---

In a JES2 system:  
/\*\$SI3-5

In a JES3 system:  
/\*\*START

---

### Communication from JCL to Operator

Use a /\*MESSAGE control statement in a JES2 system or a /\*\*OPERATOR control statement in a JES3 system to send a message to the operator when JES reads the job from the input stream. Note that the message is not synchronized with the execution of any job or step.

#### *Examples*

---

In a JES2 system:  
/\*MESSAGE JOB J67 IS HELD. CALL X65335 BEFORE RELEASING J67.

In a JES3 system:  
/\*\*OPERATOR JOB J67 IS HELD. CALL X65335 BEFORE RELEASING J67.

---

### Communication from JCL to Programmer

To communicate from your JCL to programmers, use comments fields or JCL /\*\*comment statements. The comments appear in the job log output listing if the JOB statement MSGLEVEL parameter requests that the statements be printed.

Use comments primarily to document your job and its resource requirements.

*Examples*


---

```

//*JOB J67 IS HELD UNTIL THE OPERATOR RELEASES IT.
//*THE OPERATOR SHOULD RELEASE J67 WHEN DISK 398
//*IS AVAILABLE.

```

---

**Communication from JCL to Program**

A processing program can require information that can vary from execution to execution. For example, the assembler and the linkage editor require that the programmer supply options and module attributes at execution. To provide information to a program, code the PARM parameter on the EXEC statement that executes the program.

To use the information, the processing program must contain instructions to retrieve the information. Retrieval of the PARM information is detailed in *Supervisor Services and Macro Instructions*.

*Examples*


---

```

//FIRST EXEC  PGM=IEV90,PARM=(OBJECT,NODECK,'LINECOUNT=50')
//LATER EXEC  PGM=HEWL,PARM='XREF,LIST,LET'

```

---

**PARM Values for IBM-Supplied Programs**

Some IBM-supplied programs allow you to select options from a set of alternatives. The PARM values are listed in the publication for the program. For many IBM-supplied programs, default values can be assigned to PARM values during system generation. That is, the installation can select an alternative or assign a fixed value. The system uses this default unless you specify another value in the PARM parameter when you execute the IBM-supplied program.

The installation should maintain a list of default values assigned during system generation.

**Communication from System to Operator**

The system sends to the operator console messages deemed to be needed by the operator.

**Messages during Volume Mounting**

In a JES3 system, the programmer can control the fetch messages that JES3 issues to the operator console for disk and tape volumes for a job. Code the FETCH parameter of the JES3 *//\*MAIN* statement to request one of the following:

- All fetch messages for all volumes to be mounted on JES3 setup devices.
- Fetch messages for volumes specified in DD statements that are named in the SETUP parameter on the JES3 *//\*MAIN* statement.

## Communication

- Fetch messages for volumes on named DD statements.
- No fetch messages.
- No fetch messages for volumes on named DD statements.

Regardless of the FETCH parameter, JES3 sends all the fetch messages to the job log.

### Examples

---

```
//*MAIN  FETCH=ALL
//*MAIN  FETCH=NONE
//*MAIN  FETCH=SETUP
//*MAIN  FETCH=(DDA,INDS,DD7)
//*MAIN  FETCH=/MYDS
```

---

### Messages when Job Exceeds Output Limit

JES3 sends the operator a warning message when the maximum output from the job exceeds a limit specified in the JES3 `//*MAIN` statement. The limit can be expressed in:

- Bytes to be spooled in the BYTES parameter
- Cards to be punched in the CARDS parameter
- Lines to be printed in the LINES parameter
- Pages to be printed in the PAGES parameter

If no limits are given on the `//*MAIN` statement, the system uses the installation default value for the job class.

**Use of Warning Messages:** One use for these parameters is during program testing. The warning message tells the operator that the the program is producing more output than expected. Perhaps the program is in an endless loop that contains instructions sending records to a printer or punch. The operator can halt the program's execution.

### Examples

---

```
//*MAIN  BYTES=(50,WARNING)
//*MAIN  CARDS=(120,WARNING)
//*MAIN  LINES=(200,WARNING)
//*MAIN  PAGES=(,WARNING)
```

---

## Communication from System to Time Sharing Userid

When you execute a background or batch job, you can ask the system to notify your time sharing userid or another userid when the job completes. Under the time sharing option (TSO), a background job is one that is entered from a terminal by a SUBMIT command or by executing a step to run TSO in the background. For more information, see *OS/VS2 TSO Command Language Reference*. A batch job is one that is entered through an input stream.



To request automatic notification, code in your JCL for the job one of the following:

- In a TSO background job in a JES2 or JES3 system, specify a userid in the JOB statement NOTIFY parameter. This userid must be attached to the system on which the job executes.
- In a TSO background job or a batch job in a JES2 system, specify a userid in a JES2 /\*NOTIFY statement and, if the userid is attached to another node, a node.
- In a batch job in a JES3 system, specify a userid in the JOB statement NOTIFY parameter and the processor for the userid in the ACMAIN parameter of the JES3 /\*MAIN statement.

### *Examples*

---

In a JES2 or JES3 system:

```
//MYJOB JOB , 'P. SECOR', NOTIFY=DN62PSS
```

In a JES2 system:

```
/*NOTIFY DN62PSS4  
/*NOFITY FARNODE.DN62PSS
```

In a JES3 system:

```
//MYJOB JOB , 'P. SECOR', NOTIFY=DN62PSS  
/*MAIN ACMAIN=2
```

---

## Communication from Time Sharing Userid to a JES3 System

In a JES3 system, the USER parameter on the JES3 /\*MAIN statement identifies the job with a TSO user. The job can be submitted through any input source, other than the internal reader, provided the installation does not force job naming conventions. USER allows the TSO userid to:

- Issue a TSO OUTPUT command to access sysout data sets from the job.
- Inquire about the status of the job or cancel it.

If the job executes on one processor and the TSO userid is attached to another processor, the ACMAIN parameter must identify the processor for the TSO userid.

### *Examples*

---

```
/*MAIN USER=J63ET91  
/*MAIN USER=JEN38TW, ACMAIN=2
```

---

## Communication

### Communication from Functional Subsystem to Programmer

The programmer can control whether a functional subsystem prints its messages in the output listing following the sysout data set it creates. For this control, code the PIMSG parameter on the OUTPUT JCL statement.

#### *Example*

---

```
//ODS3  OUTPUT PAGEDEF=IMAG4,PIMSG=YES
```

---

### Communication through Job Log

The system produces three system-managed data sets about a job. The system managed-data sets consist of:

- The job log, which is a record of job-related information for the programmer. The job log consists of:
  - The job control statements in the input stream, that is, the JCL statements and JES2 or JES3 statements.
  - Cataloged procedure statements for any procedure a job step calls.
  - Messages about job control statements.
- The job's hard-copy log, which is a record of all message traffic for the job to and from the operator console. These messages describe allocation of devices and volumes, execution and termination of job steps and the job, and disposition of data sets.
- System messages for the job.

The output class for the job log is set by the MSGCLASS parameter on the JOB statement or, if a job-level OUTPUT JCL statement contains a JESDS parameter, by the class that applies to the OUTPUT JCL statement. If no class is specified, the system uses the default class based on the input source of the job; the default is specified at JES initialization.

Printing of the job log is controlled by the following parameters:

- MSGLEVEL parameter of JOB statement
- All parameters on an OUTPUT JCL statement that contains a JESDS parameter

To prevent the job log from being printed, code one of the following:

- log subparameter in the JOB statement JES2 accounting information parameter
- NOLOG parameter on the JES3 /\*JOBPARM statement

## Examples

---

```
//JOBX JOB      , 'V. ST PIERRE', MSGLEVEL=(1,1)
//SMDS OUTPUT  JESDS=ALL, CLASS=D, COPIES=2, BURST=YES,
```

---

```
//JOBF JOB      (,,,,,,N)
/*JOBPARM NOLOG
```

---

```
//J1 JOB      1518, 'SECT. E98'
//O1 OUTPUT  JESDS=ALL
//O2 OUTPUT  JESDS=ALL, WRITER=JCLOGGER
//S1 EXEC    PGM=REPORT
```

This example requests that the three system-managed data sets be printed normally and that a copy of each be routed to an external writer named JCLOGGER.

---

```
//MYEX JOB      , 'DEPT. 28H', MSGCLASS=A
//SYSPROG OUTPUT JESDS=ALL, GROUPID=SYSPROG
//OPER  OUTPUT  JESDS=ALL, GROUPID=OPER
//USER  OUTPUT  JESDS=ALL, GROUPID=USER
//REMOTE OUTPUT  JESDS=ALL, DEST=REMOTE
//S1    EXEC    PGM=REPORT
//SYSPRINT DD    SYSOUT=A
```

This example creates four different output groups. Group SYSPROG will contain a copy of all three 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 all three system-managed data sets plus a copy of the data set for DD statement SYSPRINT: group USER is processed locally.

The system creates a fourth group with a system-generated group name. This group contains a copy of the three system-managed data sets plus a copy of the data set for DD statement SYSPRINT; this group is processed remotely at destination REMOTE.

---

## Printing Job Log and Sysout Data Sets Together

To print the job log and the sysout data sets from a job on the same output listing, place them in the same output class. 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.

## Communication

Or, use an OUTPUT JCL statement with a JESDS parameter to control printing of the system-managed data sets. Note that care is needed in specifying the OUTPUT JESDS statement and the sysout DD statement because:

- Any values on the sysout DD statement override those on the OUTPUT JCL statement.
- The values on the OUTPUT JCL statement **always** apply to the system-managed data sets.

Therefore, the output parameters used to process the system-managed output data sets and sysout data sets can be different, even when the data sets all reference the same OUTPUT JCL statement. For example, if the sysout DD statement specifies one output class and the JESDS statement specifies another output class, the sysout data set and system-managed data sets are placed in different subgroups and each is printed in its own output class.

### Examples

---

```
//J1  JOB  DF16,MSGCLASS=B
//S1  EXEC PGM=ABC
//OUT DD  SYSOUT=*

//J2  JOB  ,'V. FOTI',MSGCLASS=C
//S1  EXEC PGM=DEF
//OUT DD  SYSOUT=C

//J3  JOB  ,'G. ROY',MSGCLASS=D
//S1  EXEC PGM=GHI
//OT1 OUTPUT CLASS=*
//DS1 DD  SYSOUT=(,),OUTPUT=* .OT1

//J4  JOB  ,'T. POLAKOWSKI',MSGCLASS=E
//S1  EXEC PGM=JKL
//OT1 OUTPUT DEFAULT=YES,CLASS=E
//DS1 DD  SYSOUT=(,)
```

---

```
//SYSDS JOB  ,'J. HIGGINS',MSGCLASS=A
//OUT1  OUTPUT JESDS=ALL,GROUPID=JOINT,DEFAULT=YES
//STEP1 EXEC  PGM=REPORT
//REQPRT DD  SYSOUT=A
```

This example shows how to combine sysout data sets and system-managed output data sets in one output group. The system prints sysout data set REQPRT and all three system-managed data sets in the same group.

---

## Chapter 8. Protection

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Protection					
through RACF	GROUP PASSWORD USER				

Figure 8-1. Protection Task for Entering Jobs

### Protection through RACF

The IBM Resource Access Control Facility (RACF) is a program product that helps installations achieve data security by controlling access to data sets. For more information about RACF, see *Resource Access Control Facility (RACF) Security Administrator's Guide*.

For RACF protection, the user must supply to RACF a userid, a password, and, optionally, a group name. If RACF Early Verification is installed and depending on the installation's RACF options, they can be supplied in the USER, PASSWORD, and GROUP parameters on the JOB statement or, for jobs submitted by a TSO user, they can be obtained from the TSO logon.

In any RACF installation, the USER, the PASSWORD, and, optionally, the GROUP parameters are always required on JOB statements 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.

#### Examples

---

```
//MYJOB JOB D58,SUE,USER=D58STW,PASSWORD=41168X
//YOURS JOB D58,DON,USER=DSCHOF,PASSWORD=404632,GROUP=D58DISK
```

---



## Chapter 9. Resource Control

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Resource control					
of program library			JOBLIB DD STEPLIB DD DD defining PDS member		
of procedure library				PROCLIB on /*JOBPARM	PROC and UPDATE on /*MAIN
of address space	REGION ADDRSPC	REGION ADDRSPC			LREGION on /*MAIN
of processor				SYSAFF on /*JOBPARM	SYSTEM on /*MAIN
of spool partition					SPART and TRKGRPS on /*MAIN

Figure 9-1. Resource Control Task for Entering Jobs

### Resource Control of Program Library

To be executed, a program must be in one of the following libraries:

- System library
- Private library
- Temporary library

A library is a partitioned data set (PDS) on direct access storage. A PDS is divided into partitions, called members. In a library PDS, each member contains a program or part of a program. A PDS contains a list of its members, called a directory. The system uses the directory to locate a program in the library.

For details on creating and on adding members to and deleting members from a partitioned data set, see *Data Administration Guide*.

# Resource Control

## System Library

Unless a job or step specifies a private library, the system searches for a program in the system libraries when you code:

```
//stepname EXEC PGM=program-name
```

The system looks in the libraries for a member with a name or alias that is the same as the specified *program-name*. The most used system library is SYS1.LINKLIB, which contains executable programs that have been processed by the linkage editor.

If an earlier DD statement in the job defines the program as a member of a system library, refer to that DD statement to execute the program:

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

## Private Library

In a private library, each member is an executable, user-written program. To tell the system that a program is in a private library, code a DD statement defining that library as follows:

- To define a private library to be used throughout a job, place a DD statement with the ddname JOBLIB after the JOB statement and before the first EXEC statement in the job.
- To define a library to be used in only one step, place a DD statement with the ddname STEPLIB in the step.

To execute a program from a private library, code:

```
//stepname EXEC PGM=program-name
```

The system searches for the program to be executed in the library defined by the JOBLIB or STEPLIB DD statement before searching in the system libraries.

If an earlier DD statement in the job defines the program as a member of a private library, refer to that DD statement to execute the program:

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

**Use of Private Libraries:** Private libraries are particularly useful for programs used too seldom to be needed in a system library. For example, programs that prepare quarterly sales tax reports are good candidates for a private library.

**Creating a Private Library:** To create a private library, code a JOBLIB or STEPLIB DD statement and add one or more members to it in the job. The JOBLIB library is more convenient than the STEPLIB, because the JOBLIB is available to every step in the job in order to add members or to execute already added members. The STEPLIB DD must be passed or redefined in each step that uses it.

**Adding Members to a Private Library:** To add members to a library, code a DD statement that defines the library and names the member to be added to the library.



**Example of Creating and Adding to a Private Library**

---

```
//EG      JOB  5328,'MARGARET NONNSEN'
//JOBLIB DD  DSNAME=GROUPLIB,DISP=(NEW,CATLG),
//          UNIT=3350,VOL=SER=727104,
//          SPACE=(CYL,(50,3,4))
//STEP1   EXEC PGM=FINDD
//ADDPGMD DD  DSNAME=GROUPLIB(RATE),DISP=MOD,
//          VOL=REF=*.JOBLIB
//STEP2   EXEC PGM=RATE
```

In this example, the JOBLIB DD statement creates a library named GROUPLIB. STEP1 adds the program RATE to the library. STEP2 calls the program RATE.

In STEP1, the system looks for the program named FIND in SYS1.LINKLIB, because 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 JOBLIB library.

---

**Retrieving an Existing Private Library:** If several programs for a job are in the same private library, identify the library on a JOBLIB DD statement. The library is available in every step of the job for which you do not code a STEPLIB DD statement.

To make a library available to a single step, identify the library on a STEPLIB DD statement. The STEPLIB library is available only to the step that contains the STEPLIB DD statement, unless you pass the library and retrieve it in a subsequent step.

The system searches for a program in the private library you identify. If a job contains a JOBLIB DD statement and a step contains a STEPLIB DD statement, the system searches for the step's program first in the STEPLIB library and then in the system libraries. The system ignores the JOBLIB library for that step.

For a step in a job using a JOBLIB library, if you want the system libraries searched rather than the JOBLIB, code a STEPLIB DD statement that identifies a system library:

```
//STEPLIB DD  DSNAME=SYS1.LINKLIB,DISP=SHR
```

**Example of Retrieving Job and Step Libraries**

---

```
//MYJOB   JOB  MSGLEVEL=1
//JOBLIB  DD  DSNAME=LIB5.GRP4,DISP=SHR
//STEP1   EXEC PGM=FINDD
//STEP2   EXEC PGM=GATHER
//STEPLIB DD  DSNAME=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 FIND.
  - In STEP2, the system searches the library named ACCOUNTS, defined on the STEPLIB DD statement, for the program named GATHER.
-

## Resource Control

**Concatenating Private Libraries:** If a job uses programs from several libraries, you can concatenate these libraries to a JOBLIB DD statement or a STEPLIB DD statement; all the libraries being concatenated must be existing libraries. Omit the ddname from all the DD statements for the libraries, except the first.

The system searches the libraries for the program in the same order as the DD statements.

### *Example of Concatenated Libraries*

---

```
//JOBLIB DD DSNAME=D58.LIB12,DISP=(SHR,PASS)
//          DD DSNAME=D90.BROWN,DISP=(SHR,PASS),
//          UNIT=3330,VOL=SER=411731
//          DD DSNAME=A03.EDUC,DISP=(SHR,PASS)
```

---

### Temporary Library

Temporary libraries are partitioned data sets created to store a program until it is used in a later step *of the same job*. A temporary library is created and deleted within a job.

When testing a newly written program, a temporary library is particularly useful for storing the load module from the linkage editor until it is executed by a later job step. Because the module will not be needed by other jobs until it is fully tested, it should not be stored in a system library.

While the system assigns the module a name in the temporary library, the name cannot be predicted. Therefore, use the PGM parameter to identify the program by location rather than by name. Code a backward reference to the DD statement that defines the temporary library:

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

**Creating a Temporary Library:** In the step that produces the program, code a DD statement that creates a partitioned data set and place the program in it. A later step can then retrieve this program. Alternatively, you can use the virtual I/O (VIO) facilities to define a temporary library. See "Allocation of Virtual I/O" on page 15-24 for details.

### *Example*

---

```
//STEP2 EXEC PGM=IEWL
.
.
//SYSLMOD DD DSNAME=&&PARTDS(PROG),UNIT=3350,
//          DISP=(NEW,PASS),SPACE=(1024,(50,20,1))
//STEP3 EXEC PGM=*.STEP2.SYSLMOD
```

STEP2 calls the program IEWL, which link edits object modules to form a load module that can be executed. STEP2 places the module in the library defined in the SYSLMOD DD statement.

STEP3 calls the program by naming the step that created the library and 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.

---

## Resource Control of Procedure Library

Procedure libraries are partitioned data sets consisting of members that contain procedures. To call and execute a procedure cataloged in a library, code:

```
//stepname EXEC PROC=procedure-name
```

The name of the cataloged procedure is its member name or alias in the library.

If a job does not specify a procedure library, the system retrieves all cataloged procedures called by EXEC statements from the procedure libraries defined by the installation for the job's job class.

If a job's cataloged procedures are contained in another procedure library, use the following parameters to direct the system to that library. The parameters must specify procedure libraries defined during JES initialization.

- In a JES2 system, code a PROCLIB parameter on the JES2 /\*JOBPARM statement.
- In a JES3 system, code a PROC parameter on the JES3 /\*MAIN statement.

**Updating Procedure Library:** To add a procedure to an installation-defined procedure library or to modify permanently a procedure in a library, use the IEBUPDTE utility program. If modifying, tell the system operator to delay any jobs that would use the procedure during modification.

In a JES3 system, you can specify UPDATE on the JES3 /\*MAIN statement to update a procedure library. This parameter causes all jobs using the identified data set and any concatenated data sets to be held until the update is complete.

### Examples

---

In a JES2 system:

```
//JOB87 JOB , 'S. WIESENTHAL'
/*JOBPARM PROCLIB=PROC15
//S1 EXEC PROC=ALEG
//INDS DD *
      .
      (data)
      .
/*
```

In a JES3 system:

```
//JOB87 JOB , 'S. WIESENTHAL'
/*MAIN PROC=15
//S1 EXEC PROC=ALEG
//INDS DD *
      .
      (data)
      .
/*
```

In these examples, the system obtains the procedure ALEG from the procedure library PROC15.

---

# Resource Control

## Resource Control of Address Space

### Types of Storage

In MVS, the storage available for a program is virtual or real:

- **Virtual storage** is addressable space that appears to the user as real storage. Instructions and data are mapped from virtual storage into real storage locations, where they are executed.
- **Real storage** is the storage from which the processor can directly obtain instructions and data and to which it can directly return results.

**Virtual Storage:** The virtual storage address space is 4 billion bytes. The address space contains the commonly addressable system storage, the nucleus, and the private address space, which includes the user's region.

When a program is selected, the system brings it into virtual storage and divides it into pages of 4K bytes. The system transfers the pages of a program into real storage for execution and out to auxiliary storage when not needed. Paging is done automatically; to the programmer, the entire program appears to occupy contiguous space in real storage at all times. Actually, not all pages of a program are necessarily in real storage at one time. Also, the pages that are in real storage do not necessarily occupy contiguous space.

**Real Storage:** Certain programs must have all their pages in contiguous real storage while they are executing. They cannot be paged. These programs must be put into an area of virtual storage called the nonpageable dynamic area, whose virtual addresses are identical to real addresses.

Such programs include:

- Programs that modify a channel program while it is active.
- Programs that are highly dependent on time.

Such programs are the only ones for which you should request real storage. To request real storage, code `ADDRSPC=REAL` on the `JOB` or `EXEC` statement and request the amount of real storage needed in a `REGION` parameter.

### Requesting Amount and Type of Storage

The amount of space needed by a job or step can be specified in the `REGION` parameter of the `JOB` or `EXEC` statement. If `REGION` is on the `JOB` statement, each step of the job executes in the requested amount of space. If on the `EXEC` statements in a job, each step executes in its own amount of space. Use the `EXEC` statement `REGION` parameters when different steps need greatly different amounts of space.

The `REGION` parameter differs depending on whether the program uses virtual or real storage.

**Region Size for Virtual Storage:** When ADDRSPC=VIRT is coded or implied, the system establishes two values from the REGION parameter or the installation-defined default. These values are:

- An upper boundary to limit region size for variable-length GETMAINS.
- A second limiting value set by the IBM- or installation-supplied routine, IEFUSI. The system uses this second value to limit:
  - 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 a variable-length GETMAIN or the amount requested for a fixed-length GETMAIN exceeds this second value, the job or step abnormally terminates. See *SPL: System Modifications and Supervisor Services and Macro Instructions*.

The amount of space requested must include the following:

- Space for all programs to be executed.
- All additional space the programs request with GETMAIN macro instructions during execution.
- Enough unallocated space for task termination. Task termination invokes certain system components that can issue GETMAIN macro instructions for space in the user's region.

**Region Size for Real Storage:** When ADDRSPC=REAL is coded, the system establishes one value from the REGION parameter or the installation-defined default. The value is used as an upper boundary to limit region size for all GETMAINS.

The minimum region size must be:

- 8K if the program to be executed is reenterable and resides in an authorized library.
- 12K for all other programs.

Note that this is the minimum region for successful execution, but not necessarily the minimum region size for successful job completion. Programs executed in real storage should perform as much clean-up as possible before terminating.

## Resource Control

### *Examples*

---

```
//J28 JOB  , 'F. GOLAZESKI', CLASS=D
//S1 EXEC PGM=PROGREAL, REGION=20K, ADDRSPC=REAL
//DD1 DD  DSNNAME=A.B.C, DISP=OLD
//S2 EXEC PGM=PROGVIRT, REGION=75K, ADDRSPC=VIRT
//DD2 DD  DSNNAME=MYDS2, DISP=OLD
```

This example shows how to request storage for a program that must not be paged and for a program that can be paged. Step S1 executes in real storage, without paging, while step S2 executes in virtual storage, with paging.

---

```
//STEPA EXEC PROC=MYPROC8, REGION.FIRST=750K,
//          REGION.SECOND=700K
```

This EXEC statement assigns space requests to two procedure steps, FIRST and SECOND, of a procedure named MYPROC8.

---

### Requesting Amount of Logical Storage in a JES3 System

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 value to improve job scheduling. For more information, see *JES3 SPL: Initialization and Tuning*.

Use LREGION carefully. If the values selected for LREGION are too small, the job may take longer to run.

### *Example*

---

```
//*MAIN LREGION=100K
```

---

## Resource Control of the Processor

### Selecting a Processor in JES2

In a JES2 multi-access spool configuration, jobs enter from local input streams, from remote work stations, and from processors at other network nodes. If an entering job does not specify a system, JES2 can assign the job to execute on any system in the configuration.

In a multi-access spool configuration, a job can request execution on specific systems. This request is made by coding:

```
/*JOBPARM SYSAFF=cccc
/*JOBPARM SYSAFF=(cccc,cccc,cccc)
/*JOBPARM SYSAFF=*
/*JOBPARM SYSAFF=ANY
```

A specified system processes the job's JCL and executes the job. The output from the job can be processed by any system in the multi-access spool configuration.

You should request a specific system when a job has special processing requirements not available on all systems in the configuration. For example, an emulation job might need to run on a particular system.

For more information on the JES2 multi-access spool configuration, see *SPL: JES2 Initialization and Tuning*.

**Independent Mode:** If the job needs to be processed by a system in independent mode, code:

```
/*JOBPARM SYSAFF=(cccc,IND)
/*JOBPARM SYSAFF=(,IND)
/*JOBPARM SYSAFF=(ANY,IND)
```

A specified system, provided it is operating in independent mode, processes the job's JCL and executes the job. The same system processes the job's output.

Independent mode is useful for testing new components with selected jobs while in a shared configuration.

### Examples

---

```
/*JOBPARM SYSAFF=SYS2
/*JOBPARM SYSAFF=(S333,IND)
/*JOBPARM SYSAFF=(*,IND)
```

---

## Selecting a Processor in JES3

JES3 automatically selects a processor for a job based on the resources that JES3 knows the job needs in order to execute. These resources are:

- Devices
- Volumes
- Data sets
- Processor features, such as an emulator, a nonstandard catalog, or a connection to a particular system-managed device.

If a job must have resources that JES3 does not control or that JES3 cannot infer from the job control statements, name the processor(s) that should or should not execute the job by coding:

```
//*MAIN SYSTEM=ANY
//*MAIN SYSTEM=JGLOBAL
//*MAIN SYSTEM=JLOCAL
//*MAIN SYSTEM=(main-name,main-name,...)
//*MAIN SYSTEM=/(main-name,main-name,...)
```

## Resource Control

**Relationship to Other Parameters:** The requested processor must be consistent with other parameters specified in the job control statements:

- CLASS parameter on the JOB statement or `//*MAIN` statement. A processor or processors are defined for each valid job class during JES3 initialization. If the SYSTEM parameter specifies a processor that does not execute jobs of the specified class, JES3 abnormally terminates the job.
- DD statement UNIT parameter that specifies a device-number for a device that is JES3-managed or jointly JES3/MVS managed. The specified device must be attached to the requested processor. Also, because a specific device is requested, the SYSTEM parameter is required.
- The TYPE parameter on the `//*MAIN` statement must specify the system running on the requested processor.
- The processing requests made in JES3 `//*PROCESS` statements. Any dynamic support programs called in `//*PROCESS` statements must be able to be executed on the requested processor.

### Examples

---

```
//*MAIN SYSTEM=(PRS1,PRS3)
```

---

## Resource Control of Spool Partitions in a JES3 System

When JES3 reads a job, it initially places the job on a spool volume or volumes. The spool volumes can be divided by the installation into groups, known as partitions. During JES3 initialization, partitions are defined and associated with output classes, job classes, and processors. See *SPL: JES3 Initialization and Tuning* for details.

During job processing, JES3 allocates spool data sets to a partition, as follows, in override order:

1. The spool partition for the output class of the sysout data set.
2. The spool partition for the job's class.
3. The spool partition for the processor executing the job.
4. The default spool partition.

You can use the `//*MAIN` statement to override the JES3 partition allocations, except for allocation of partitions for sysout data sets and SYSIN data sets. A sysout data set is always placed in the partition used for its output class; a SYSIN data set is always placed in the default spool partition. Depending on how the installation defines the partitions, you can make JES3 allocate all the spool data for a job or all the spool data of a particular type, such as



output, to a specified spool partition. Thus, you can limit the number of spool volumes that JES3 uses for a job's spool data sets. To control the spool partition, code:

```
//*MAIN SPART=partition-name
```

### Examples

---

```
//ONE JOB , 'PAT EGAN'  
//*MAIN SYSTEM=SY2  
//S1 EXEC PGM=ABC  
//OUT1 DD SYSOUT=N  
//OUT2 DD SYSOUT=S
```

During initialization, the installation assigned spool partitions as follows:

- PARTD is assigned to output class S.
- PARTC is assigned to processor SY2.
- PARTA is the default partition.
- No partition is assigned to output class N.

The job's input spool data sets are allocated to the default spool partition, PARTA.

Because the job executes on processor SY2 and no partition is assigned for output class N, the sysout data set OUT1 is allocated to partition PARTC.

Sysout data set OUT2 is allocated to PARTD.

---

```
//TWO JOB , 'LEE BURKET'  
//*MAIN CLASS=IMSBATCH, SYSTEM=SY2  
//S1 EXEC PGM=DEF  
//OUT1 DD SYSOUT=N  
//OUT2 DD SYSOUT=S
```

During initialization, the installation assigned spool partitions as for job ONE, with the following addition:

- PARTB is assigned to job class IMSBATCH.

The sysout data set OUT1 is allocated to partition PARTB, the job class's partition. Note that the job class's partition overrides the processor's partition.

---

```
//THREE JOB , 'T. POLAKOWSKI'  
//*MAIN CLASS=IMSBATCH, SPART=PARTE, SYSTEM=SY2  
//STEP1 EXEC PGM=GHI  
//OUT DD SYSOUT=N  
//OUT2 DD SYSOUT=S
```

During initialization, the installation assigned spool partitions as for job TWO.

The sysout data set OUT1 is allocated to partition PARTE, as specified in the SPART parameter. Note that the SPART parameter overrides the processor's partition and the job class's partition.

---



## Part 3. Tasks for Processing Jobs

This part describes how to process jobs that have been entered into the system. These tasks are all optional. They are:

- Processing control
- Performance control

### Part 3 Contents

<b>Chapter 10. Processing Control</b>	10-1
Processing Control by Terminating Execution	10-2
Bypassing or Executing Steps Based on Return Codes	10-2
Uses of Return Code Tests	10-2
Relationship of the COND Parameters on JOB and EXEC Statements	10-3
Step Execution after a Preceding Step Abnormally Terminates	10-3
Compatible Return Code Tests	10-4
Examples of JOB Statement Return Code Tests	10-5
Examples of EXEC Statement Return Code Tests	10-5
Examples of EXEC COND Parameters with EVEN and ONLY	10-6
Examples of COND Return Code Testing in a Job	10-7
Examples of COND Parameters in Procedures	10-8
Examples of COND Parameters that Force Step Execution	10-10
Cancelling Job that Exceeds Output Limit	10-10
Use in Testing	10-10
Examples	10-10
Processing Control by Timing Execution	10-10
JOB and EXEC TIME Parameter	10-11
Examples	10-11
JES2 Time Parameters	10-12
Examples	10-12
Processing Control for Testing	10-12
Altering Usual Processing for Testing	10-12
Scanning JCL for Errors	10-12
Examples	10-13

## Part 3

Using IEFBR14 Program for Testing	10-13
Considerations when Using IEFBR14	10-13
Examples	10-13
Using Nonstandard Processing	10-14
Example	10-14
Dumping after Error	10-14
Examples	10-15
<b>Chapter 11. Performance Control</b>	11-1
Performance Control by Job Class Assignment	11-2
Examples	11-2
Performance Control by Selection Priority	11-3
Priority for JES2 Jobs	11-3
Use of Priority	11-3
Examples	11-3
Priority for JES3 Jobs	11-3
Example	11-3
Priority Aging	11-4
Performance Control by Dispatching Priority	11-4
Examples	11-4
Performance Control by Performance Group Assignment	11-5
Examples	11-5
Performance Control by I/O-to-Processing Ratio in a JES3 System	11-5
Examples	11-5

Chapter 10. Processing Control

TASKS FOR PROCESSING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Processing control					
by terminating execution	COND	COND			CANCEL in BYTES, CARDS, LINES, and PAGES on /*MAIN
by timing execution	TIME or time in JOB JES2 accounting information	TIME		TIME on /*JOBPARM	
for testing: (1) by altering usual processing  (2) by dumping after error	TYPRUN CLASS	PGM = IEFBR14  PGM = JCLTEST PGM = JSTTEST (JES3 only)	SYSABEND DD SYSMDUMP DD SYSUDUMP DD  To format dump on 3800 Printing Subsystem, FCB = STD3 and CHARS = DUMP on dump DD		/*PROCESS /*ENDPROCESS  DUMP in BYTES, CARDS, LINES, and PAGES on /*MAIN

Figure 10-1. Processing Control Task for Processing Jobs

## Processing Control

### Processing Control by Terminating Execution

#### Bypassing or Executing Steps Based on Return Codes

Depending on the results of a job step, you may need to bypass or execute later steps. To indicate the results of its execution, a program can issue a *return code*. Using a COND parameter, you can test the return code and, based on the test, either bypass or execute a step. The COND parameter can be specified on either a JOB or EXEC statement by coding:

```
//jobname JOB acct,progname,COND=(code,operator)
//jobname JOB acct,progname,COND=((code,operator),(code,operator))

//stepname EXEC PGM=x,COND=(code,operator)
//stepname EXEC PGM=x,COND=(code,operator,stepname)
//stepname EXEC PROC=x,COND=((code,operator,stepname.procstepname))

//stepname EXEC PGM=x,COND=EVEN
//stepname EXEC PGM=x,COND=ONLY
//stepname EXEC PGM=x,COND=((code,operator),EVEN)
//stepname EXEC PGM=x,COND=((code,operator,stepname),ONLY)
```

If an EXEC statement COND parameter causes a step to be bypassed, only that step is not executed; the following steps are executed or not, depending on their COND parameters. If a JOB statement COND parameter causes a step to be bypassed, the system bypasses all remaining job steps.

Bypassing a step because of an EXEC COND parameter is not the same as abnormally terminating the step. Bypassing permits the following steps to be executed; abnormally terminating causes all following steps to be bypassed, unless they contain EVEN or ONLY in their EXEC COND parameters.

#### Uses of Return Code Tests

Certain IBM programs produce standard return codes. For example, a compiler or linkage editor returns a code of 8 to indicate serious errors in the compiled or link-edited program; the program may not execute correctly. Before executing a newly compiled or link-edited program, test the return code from the compiler or linkage editor; if it is 8, bypass execution of the program.

In user-written programs, assign a return code to signify a certain condition. For example, STEP1 of a job reads accounts that subsequent steps process. STEP1 sets a return code of 10 if delinquent accounts are found. STEP3 processes only delinquent accounts. Before STEP3 executes, test the return code from STEP1:

- If the return code from STEP1 is 10, indicating delinquent accounts, execute STEP3.
- If the return code from STEP1 is not 10, bypass STEP3.

## Relationship of the COND Parameters on JOB and EXEC Statements

The affect of return code tests on the different statements is:

- **The JOB statement COND parameter** performs the same return code tests for every step in a job. If a JOB statement return code test is satisfied, the job terminates.
- **An EXEC statement COND parameter** performs return code tests for only its step in a job. Using EXEC COND parameters, different tests can be performed for each step. Thus, EXEC COND parameters are useful 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 a return code.

A COND parameter on the first EXEC statement in a job is meaningless and is ignored by the system.

- **The JOB COND parameter, when EXEC statements also contain COND parameters,** performs the same return code tests for every step in the job.
  - If the JOB statement return code test is satisfied, the job terminates. The job terminates regardless of whether or not any EXEC statements contain COND parameters and whether or not an EXEC return code test would be satisfied.
  - If the JOB statement return code test is not satisfied, the system then checks the COND parameter on the EXEC statement for the next step. If the EXEC statement return code test is satisfied, the system bypasses that step and begins processing of the following step, including return code testing.

The COND parameter on both the JOB and EXEC statements is useful to set some conditions for all steps in the job and other conditions for particular steps.

- **No COND parameters on JOB or EXEC statements** means the system does not perform any return code tests, but tries to execute each step in the job.

## Step Execution after a Preceding Step Abnormally Terminates

Abnormal termination of a step usually causes the system to bypass subsequent steps and to terminate the job. However, the EXEC statement COND parameter lets you request execution of a step by coding:

```
//stepname EXEC PGM=x,COND=EVEN
```

The step is to be executed even if one or more of the preceding steps abnormally terminates. That is, the step will always be executed, whether or not a preceding step abnormally terminates.

```
//stepname EXEC PGM=x,COND=ONLY
```

The step is to be executed only if one or more of the preceding steps abnormally terminates. That is, the step will not be executed, unless a preceding step abnormally terminates.

## Processing Control

If a step abnormally terminates, the system scans the EXEC COND parameter for the next step for an EVEN or ONLY subparameter. If neither is present, the system bypasses the step. If EVEN or ONLY is specified, the system makes any requested return code tests against the return codes from previous steps that executed and did not abnormally terminate. The step is bypassed if any test is satisfied. Otherwise, the step is executed.

*Note:*

- EVEN and ONLY are ignored if a step is abnormally terminated because it exceeded the time limit for the job.
- 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.
- For the system to act on the COND parameter, the step must abnormally terminate while the program has control. If a step abnormally terminates during scheduling, due to failures such as JCL errors or inability to allocate space, the system bypasses the remaining steps, no matter what the COND parameter requests.

### Compatible Return Code Tests

The system applies the return code tests on the JOB COND parameter against the return code, if any, produced by each step in the job. To take advantage of this parameter, the return codes for each step should have compatible meanings. For example, the COBOL compiler and the linkage editor have compatible return codes:

- 4 Minor errors were found, but a compiled program or load module was produced. Execution may be successful.
- 8 Major errors were found, but a compiled program or load module was produced. Execution will probably not be successful.
- 12 Serious errors were found. A compiled program or load module was not produced.

Code the return code as follows:

**COND=(4,LT)** if you want to continue processing despite the small errors. The job terminates only if the return code of any step is greater than 4.

**COND=(4,LE)** if you want to continue processing only if no errors occur. The job terminates if the return code of any step is greater than or equal to 4.



## Examples of JOB Statement Return Code Tests

---

```
//J1 JOB , 'LEE BURKET', COND=( (10,GT), (20,LT) )
```

This example asks “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 both are false after each step executes, the system executes all job steps.

For example, if a step returns a code of 12, neither test is satisfied. The next step is executed. However, if a step returns a code of 25, the first test is false, but the second test is satisfied: 20 is less than 25. The system bypasses all remaining job steps.

---

```
//J2 JOB , 'D WEISKOPF', COND=( (50,GE), (60,LT) )
```

This example says “If 50 is greater than or equal to a return code, or 60 is less than a return code, bypass the remaining job steps.” In other words, the job continues as long as the return codes are 51 through 60.

---

```
//J3 JOB , 'E. SASSMANN', COND=( 8,NE )
```

This example shows one return code test.

---

```
//J4 JOB COND=( (5,GT), (8,EQ), (12,EQ), (17,EQ), (19,EQ), (21,EQ), (23,LE) )
```

This example shows the maximum of eight return code tests. The job continues *only* if the return codes are: 5, 6, 7, 9, 10, 11, 13, 14, 15, 16, 18, 20, or 22.

---

## Examples of EXEC Statement Return Code Tests

---

```
//S3 EXEC PGM=U, COND=( (20,GT,STEP1), (60,EQ,STEP2) )
```

This example says “Bypass this step if 20 is greater than the return code STEP1 issues, or if STEP2 issues a return code of 60.”

---

```
//S4 EXEC PGM=V, COND=( (20,GT,STEP1), (60,EQ) )
```

This example says “Bypass this step if 20 is greater than the return code STEP1 issues, or if any preceding step issues a return code of 60.”

---

```
//T7 EXEC PGM=B15, COND=( 10,LT )  
//STEP8 EXEC PGM=MYPROG, COND=( 15,NE,STEP5 )
```

These examples show single return code tests.

---

```
//NEXT EXEC PGM=AFTERPRC, COND=( 7,LT,STEP4.LINK )
```

This example says “Bypass this step if 7 is less than the return code issued by a procedure step named LINK in the cataloged procedure called by the EXEC statement named STEP4.”

---

# Processing Control

## Examples of EXEC COND Parameters with EVEN and ONLY

---

```
//S5 EXEC PGM=R,COND=EVEN  
//R8 EXEC PGM=S,COND=((5,LT),EVEN)  
//S6 EXEC PGM=T,COND=ONLY  
//CX EXEC PGM=U,COND=((4,GE,STEP3),(8,EQ,STEP2),ONLY,(12,LT,BX))
```

---

```
//LATE EXEC PGM=CLEANUP,COND=EVEN
```

This example says "Execute program CLEANUP even if one or more of the preceding steps abnormally terminated."

---

```
//LATER EXEC PGM=SCRUB,COND=((10,LT,STEPA),(20,EQ),ONLY)
```

This example says "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."

---

```
//LATEST EXEC PGM=FIX,COND=((10,LT,STEPA),(20,EQ),EVEN)
```

This example says "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."

---

```
//EXG EXEC PGM=A1,COND=(EVEN,(4,GT,STEP3))  
//EXH EXEC PGM=A2,COND=((8,GE,STEP1),(16,GE),ONLY)  
//EXI EXEC PGM=A3,COND=((15,GT,STEP4),EVEN,(30,EQ,STEP7))
```

---

## Examples of COND Return Code Testing in a Job

<i>Input Stream</i>	<i>RC</i>	<i>Tests Performed</i>
//MYJOB JOB ,A.SMITH,COND=(10,LT) //STEP1 EXEC PGM=A . . .	6	Before STEP2: 1. Is 10 less than 6? No. 2. Is the return code 2 or 4? No. Execute STEP2
//STEP2 EXEC PGM=B,COND=((2,EQ),(4,EQ)) . . .	2	Before STEP3: 1. Is 10 less than 2 or 6? No. 2. Did one or more of the preceding steps terminate abnormally? No. Bypass STEP3.
//STEP3 EXEC PGM=C,COND=ONLY . . .	-	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.
//STEP4 EXEC PGM=D, // COND=((5,GT,STEP1),(2,EQ)) . . .	-	Before STEP5: 1. Is 10 less than 2 or 6? No. Execute STEP5.
//STEP5 EXEC PGM=E . . .	9	Before STEP6: 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.
//STEP6 EXEC PGM=F, // COND=((8,GT,STEP5),EVEN) . . .	10	Before STEP7: 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.
//STEP7 EXEC PGM=G,COND=(4,GT,STEP4) . . .	12	Before STEP8: 1. Is 10 less than 12, 10, 9, 2, or 6? Yes. Bypass STEP8 and STEP9.
//STEP8 EXEC PGM=H . . .	-	
//STEP9 EXEC PGM=I,COND=ONLY	-	

# Processing Control

---

## Input Stream

```
//ABC JOB 12345,COND=(5,EQ)
//STEP1 EXEC PGM=A
.
.
//STEP2 EXEC PGM=B,COND=(7,LT)
.
.
//STEP3 EXEC PGM=C,
// COND=((20,GT,STEP1),EVEN)
.
.
//STEP4 EXEC PGM=D,COND=((3,EQ),ONLY)
.
.
//STEP5 EXEC PGM=E,COND=(2,LT,STEP3)
.
.
//STEP6 EXEC PGM=F
.
.
//STEP7 EXEC PGM=G,
// COND=((6,EQ,STEP5),ONLY)
.
.
//STEP8 EXEC PGM=H,COND=EVEN
.
.
//STEP9 EXEC PGM=I
```

## RC

4

ABEND

6

5

-

-

## Tests Performed

Before STEP2:

1. Is 5 equal to 4? No.
2. Is 7 less than 4? No. Execute STEP2.

Before STEP3:

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:

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:

1. Is 5 equal to 5, 6, or 4? Yes. Bypass STEP8 and STEP9.

---

## Examples of COND Parameters in Procedures

```
//TEST EXEC PROC=PROC4,COND.STEP4=((7,LT,STEP1),
// (5,EQ),EVEN),COND.STEP6=((2,EQ),
// (10,GT,STEP4))
```

In this example, the EXEC statement that calls procedure PROC4 passes COND parameters to two steps, STEP4 and STEP6,

---

---

```
//TEST EXEC PROC=MYPROC,COND=((7,LT,STEP1),(5,EQ))
```

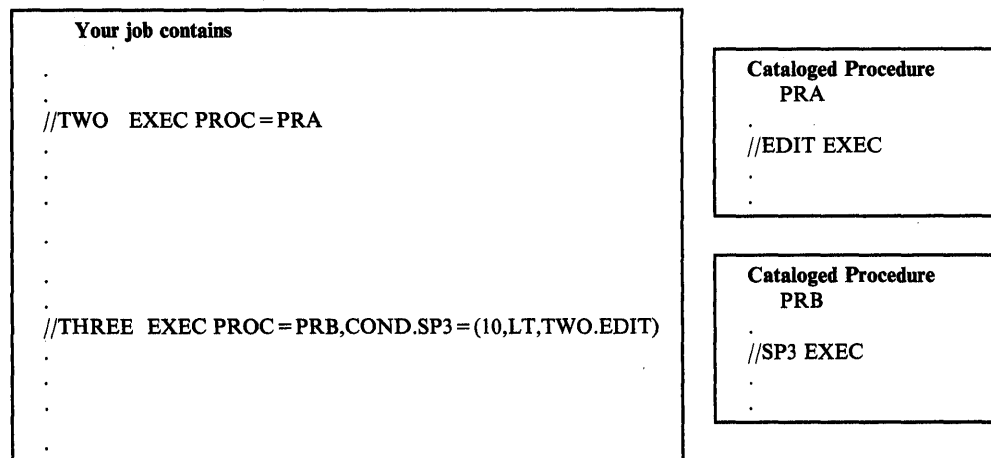
This EXEC statement establishes a COND parameter for all steps in the called procedure. It overrides any COND parameters in the procedure, if coded.

---

```
//PS3 EXEC PGM=ADD3,COND=(5,EQ,STEP2)
```

In this EXEC statement in a procedure, STEP2 in the COND parameter can be the name of either a preceding step in the procedure or of a preceding step in the job.

---



This example shows a procedure EXEC statement COND parameter that tests the return code from a step in another procedure called by a previous step in this job.

1. Step TWO calls cataloged procedure PRA, which contains procedure step EDIT. The system is to test the return code from EDIT.
2. Step THREE calls cataloged procedure PRB, which contains procedure step SP3. Execution of SP3 should depend on the return code from EDIT.
3. The COND parameter in EXEC statement THREE directs the system to bypass SP3 if 10 is less than the return code from procedure step EDIT.

The COND parameter could also have appeared on EXEC statement SP3:

```
//SP3 EXEC PGM=DEPEND,COND=(10,LT,TWO,EDIT)
```

To direct the system to bypass all steps in procedure PRB, code the COND parameter without the SP3 qualifier, as follows:

```
//THREE EXEC PRB,COND=(10,LT,TWO.EDIT)
```

---

## Processing Control

### Examples of COND Parameters that Force Step Execution

---

```
//S1 EXEC PGM=A  
.  
.  
//CLEANUP EXEC PGM=FIX,COND=((12,LT,S1),(12,GT,S1))
```

In this example, you force step CLEANUP to execute if step S1 executes but issues a return code of 12 to indicate that data sets might contain invalid records. The program FIX would clean up the invalid records.

---

### Cancelling Job that Exceeds Output Limit

JES3 cancels a job when the maximum output from the job exceeds a limit specified in the JES3 `//*MAIN` statement. The limit can be expressed in:

- Bytes to be spooled in the BYTES parameter
- Cards to be punched in the CARDS parameter
- Lines to be printed in the LINES parameter
- Pages to be printed in the PAGES parameter

If no limits are given on the `//*MAIN` statement, the system uses the installation default value for the job class.

**Use in Testing:** One use for these parameters is during program testing. These parameters can cancel a program that is in an endless loop that contains instructions sending records to a sysout data set.

### Examples

---

```
//*MAIN BYTES=(50,CANCEL)  
//*MAIN CARDS=(120,CANCEL)  
//*MAIN LINES=(200,CANCEL)  
//*MAIN PAGES=(,CANCEL)
```

---

## Processing Control by Timing Execution

To control processing based on the processor time needed to execute a program, code one of the following time parameters:

```
//jobname JOB acct,programe,TIME=value  
//stepname EXEC PGM=x,TIME=value  
//jobname JOB (,time)  
/*JOBPARM TIME=value
```

## JOB and EXEC TIME Parameter

The TIME parameter on the JOB or EXEC statement specifies the maximum length of time a job or step is to use the processor. Two benefits of the TIME parameter are:

- The system prints the actual processor time used by the job or step in the messages in the job log.
- When a job or step exceeds the maximum time, the system abnormally terminates it or gives control to a user exit routine established through System Management Facilities (SMF). Thus, the TIME value limits the processor time wasted by a looping program.

By coding TIME=1440, the TIME parameter can instead be used to give a job or step an unlimited amount of time. Specifically, the system allows a step to remain in a continuous wait state for an unlimited time, rather than the time limit established through SMF.

Because the processor time-used field is checked at intervals of about 10.5 seconds, the actual amount of time that a job or step uses the processor can exceed the time specified on the TIME parameter by up to 10.5 seconds.

### *Examples*

---

```
//FIRST JOB      , 'E.D. WILLIAMSON' , TIME=2  
//STEP1 EXEC    PGM=A, TIME=1  
//STEP2 EXEC    PGM=B, TIME=1
```

In this example, the job is allowed 2 minutes of execution time and each step is allowed 1 minute. Should either step try to execute beyond 1 minute, the job will terminate beginning with that step.

---

```
//SECOND JOB     , 'M. CARLO' , TIME=3  
//STEP1 EXEC    PGM=C, TIME=2  
//STEP2 EXEC    PGM=D, TIME=2
```

In this example, the job is allowed 3 minutes of execution time. Each step is allowed 2 minutes of execution time. Should either step try to execute beyond 2 minutes, the job will terminate beginning with that step. If STEP1 executes in 1.74 minutes and if STEP2 tries to execute beyond 1.26 minutes, the job will be terminated because of the 3-minute time limit specified on the JOB statement.

---

```
//AAA EXEC      PROC=PROC5, TIME=20
```

This EXEC statement sets a time limit for an entire procedure. This specification overrides any TIME parameters in the procedure, if coded.

---

```
//AAA EXEC      PROC=PROC5, TIME.ABC=20, TIME.DEF=(3,40)
```

This EXEC statement sets a time limit for two steps, ABC and DEF, of the called cataloged procedure.

---

# Processing Control

## JES2 Time Parameters

In a JES2 system, you can code a time value in the JES2 format accounting information parameter on the JOB statement or in a TIME parameter on the JES2 /\*JOBPARM statement. If the job execution time exceeds this value, JES2 sends a message to the operator.

### *Examples*

---

```
//J3 JOB (,,3)
/*JOBPARM TIME=3
```

---

Both of these statements specify that the job cannot use the processor for more than 3 minutes.

---

# Processing Control for Testing

## Altering Usual Processing for Testing

### Scanning JCL for Errors

Before using a new set of job control statements, you can ask the system to scan them for syntax errors without executing any steps or allocating any devices. To do this scanning, code:

- For a job in a JES2 or JES3 system:

```
//jobname JOB acct,progname, TYPRUN=SCAN
```

- For a job in a JES2 system, where x is a class defined during JES2 initialization to force job control statement scanning:

```
//jobname JOB acct,progname, CLASS=x
```

- For a step in a JES3 system:

```
//stepname EXEC PGM=JCLTEST
//stepname EXEC PGM=JSTTEST
```

The system scans for:

- Invalid keywords.
- Invalid characters.
- Parentheses errors.
- In a JES3 system only, parameter value errors or excessive parameters.
- Invalid syntax on statements in cataloged procedures invoked by any scanned EXEC statements.



The system does not check for misplaced statements or the syntax of subparameters of JCL parameters.

## *Examples*

---

```
//JB16 JOB      , 'M. CARLO', TYPRUN=SCAN
//TG  JOB      RK988, SMITH, CLASS=S
//S1  EXEC     PGM=JCLTEST
//S2  EXEC     PGM=JSTTEST
```

---

## Using IEFBR14 Program for Testing

IEFBR14 is a two-line program that clears register 15, thus passing a return code of 0, and then branches to the address in register 14, which returns control to the system. If a step requests IEFBR14 instead of the program that the JCL actually supports, the system does the following:

- Checks all the job control statements in the step for syntax.
- Allocates direct access space for data sets.
- Performs data set dispositions.

To test with IEFBR14, substitute IEFBR14 for the name of the program, as follows:

```
//stepname EXEC PGM=IEFBR14,...
```

**Considerations when Using IEFBR14:** Although the system allocates space for data sets, it does not initialize the data sets. Therefore, any attempt to read from one of these data sets will produce unpredictable results. Also, IBM does not recommend allocation of multi-volume data sets while executing IEFBR14.

If you created a data set when testing with IEFBR14, the data set's status in the DD DISP parameter is old when you execute the actual program.

Because IEFBR14 does not open any data sets, a DD DISP parameter of CATLG does not make the system catalog a data set, if one of the following is true:

- The DD statement requested a nonspecific tape volume.
- The DD statement requested a tape volume with dual density options, but the DCB DEN subparameter did not specify the density.

When executing IEFBR14, if a DD DISP parameter specifies CATLG or UNCATLG, the system issues an operator message to mount the volume. If it is not necessary to mount the volume, code DEFER on the UNIT parameter of the DD statement.

## *Examples*

---

```
For testing:
//STEP1 EXEC PGM=IEFBR14, COND=(8, LE), TIME=2
```

```
For executing after testing:
//STEP1 EXEC PGM=WKLYRPT, COND=(8, LE), TIME=2
```

---

# Processing Control

## Using Nonstandard Processing

In a JES3 system, you can use nonstandard job processing in testing. Standard job processing consists of the following standard scheduler functions:

- Converter/interpreter service
- Main service
- Output service
- Purge service

A nonstandard job uses one or more special processing functions in place of or in addition to the standard functions or skips one or more standard functions. Specify nonstandard processing by following the JOB statement with a JES3 `//*PROCESS` statement for each processing function to be performed.

End the `//*PROCESS` statements with a `//*ENDPROCESS` statement or a JCL statement.

### *Example*

---

```
//TESTA    JOB    , 'E. HARMANTAS'
//*PROCESS CI
//STEP1    EXEC  PGM=NEWPROG
//DD28     DD    SYSOUT=A
//DD29     DD    *
           .
           .
           (data)
           .
/*
```

This example asks for only the converter/interpreter service, CI. The converter/interpreter scans the job's syntax for errors. The program will not be executed or the job's output processed. However, the job will be purged from the system.

---

## Dumping after Error

To request that the system dump the storage occupied by a failing program and other storage needed to debug the program, code one of the following:

- `SYSABEND`, `SYSMDUMP`, or `SYSUDUMP DD` statement in the step to be dumped. The system produces the requested dump if the step terminates abnormally or if the step starts to terminate abnormally, but the system recovery procedures allow the step to terminate normally.
- `DUMP` in the `BYTES`, `CARDS`, `LINES`, or `PAGES` parameter on the JES3 `//*MAIN` statement in the job and a `SYSABEND`, `SYSMDUMP`, or `SYSUDUMP DD` statement in the step to be dumped. The system produces the dump requested by the dump DD statement if JES3 cancels the job because the maximum output exceeds the `BYTES`, `CARDS`, `LINES`, or `PAGES` limit or, if no limits are given, the installation default limit for the job class.

If the dump is to be printed directly on a 3800 Printing Subsystem, the SYSABEND or SYSUDUMP DD statement can request a high-density dump by specifying:

- FCB=STD3 to produce dump output at 8 lines per inch.
- CHARS=DUMP to produce 204-character print lines.

### Examples

---

```
//S1      EXEC PGM=TESTING
//DS1     DD  SYSOUT=C
//SYSABEND DD  SYSOUT=A,FCB=STD3,CHARS=DUMP
//INDS    DD  *
          .
          .
          (data)
          .
/*
```

This example produces a high-density dump, if TESTING abnormally terminates.

---

```
//J3JB     JOB  , 'J.T. HIGGINS' ,MSGCLASS=B
//*MAIN    LINES=(50,DUMP)
//S1      EXEC PGM=OLDPROG
          .
          .
          .
//S2      EXEC PGM=NEWPROG
//SYSUDUMP DD  SYSOUT=D
          .
          .
          .
```

If the first step exceeds 50,000 lines of output, JES3 cancels the job but does not write a dump because the first step does not contain a dump DD statement. If the combined output from S1 and S2 exceeds 50,000 lines, JES3 cancels the job and writes a SYSUDUMP dump to the sysout data set for class D.

---



## Chapter 11. Performance Control

TASKS FOR PROCESSING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Performance control					
by job class assignment	CLASS				CLASS on /**MAIN
by selection priority	PRTY			/**PRIORITY	
by dispatching priority		DPRTY			
by performance group assignment	PERFORM				
by I/O-to-processing ratio					IORATE on /**MAIN

Figure 11-1. Performance Control Task for Processing Jobs

## Performance Control

### Performance Control by Job Class Assignment

The system can balance the mix of jobs being executed based on the class and priority assigned to each job. The installation should assign classes and priorities so that the system will not simultaneously execute jobs that compete for the same resources.

A JES2 installation can have 36 job classes; a JES3 installation can have 255 job classes. Two more classes are reserved for started tasks and time sharing users. The installation arbitrarily determines the type of job to be placed in each class. In general, all jobs with the same characteristics should be in the same class. Then the installation assigns to each of its initiators, which are the system components that start jobs executing, one or more classes of jobs that the initiator can start.

For example, an installation can identify separate classes for the following job types:

- I/O-bound jobs.
- Processor-bound jobs.
- Jobs being debugged.
- Jobs using a particular resource.

Using these example job classes, the installation can assign job classes to initiators so that:

- I/O-bound jobs will execute at the same time as processor-bound jobs. This multiprogramming helps both types of jobs complete faster.
- All programs that use tape drives will be in the same class, if the installation contains only a few tape drives. If this class can be started by only one initiator, programs needing tapes will not be executed at the same time.
- All programs that use a data base will be in the same class, if the data base must be accessed serially. If this class can be started by only one initiator, programs accessing the data base will execute serially.

The installation should maintain a list of job classes and the types of jobs to be assigned to them.

In a JES2 system, assign a job to a job class by coding:

```
//jobname JOB acct,progname,CLASS=x
```

In a JES3 system, assign a job to a job class, which is part of a job class group, by coding either of the following:

```
//jobname JOB acct,progname,CLASS=x  
//*MAIN CLASS=x
```

#### *Examples*

---

```
//MYJOB JOB ACCT24,BIRDSALL,CLASS=F  
//*MAIN CLASS=H
```

---

## Performance Control by Selection Priority

Within a JES2 job class or a JES3 job class group, the system selects jobs for execution in order by priority. The higher the priority number, the sooner the job is selected. Jobs with the same priority are selected on a first-in first-out basis.

### Priority for JES2 Jobs

If a priority is not specified, JES2 uses installation algorithms to calculate the job's priority based on the execution time and the estimated amount of output. The operator can assign a different priority or you can code one of the following:

```
//jobname JOB acct,progname,PTY=x
/*PRIORITY x
```

JES2 also uses the execution time and output amount to monitor job execution time and output. If you do not code these estimates, JES2 assumes installation defaults. If your job exceeds the coded or assumed estimates, JES2 issues warning messages to the operator or cancels the job, with or without a dump.

Following job execution, JES2 sets a job's selection priority to:

1 if the job's execution priority was 12 or less  
15 if the job's execution priority was 13, 14, or 15

**Use of Priority:** An installation can specify that jobs with shorter execution times and less output should be assigned higher priorities. To make sure that programmers specify correct times and output, the installation can instruct the operator to cancel jobs that exceed the estimates.

#### Examples

---

```
//JOB10 JOB , 'FLO JONES',PTY=14
/*PRIORITY 14
```

---

### Priority for JES3 Jobs

To assign a priority to your job, you can code the following:

```
//jobname JOB acct,progname,PTY=x
```

The operator can change a job's priority; see *JES3 Commands*.

#### Example

---

```
//JOB10 JOB , 'FLO JONES',PTY=14
```

---

# Performance Control

## Priority Aging

JES2 increases the priority of a job as it waits to be executed in the system. JES2 keeps raising the job's priority until it is executed.

JES3 increases a job's priority based on the number of times the job is passed over for selection. A job can be passed over because not enough devices are available or because another job has a needed volume or data set or because not enough storage is available.

The installation defines priority aging; you cannot specify it using JCL.

## Performance Control by Dispatching Priority

For most steps, the job's dispatching priority should default to the automatic priority group (APG) priority. The APG is an algorithm that the system resources manager (SRM) uses to increase system throughput by dynamically adjusting the dispatching priority of address spaces.

However, for some steps, you will want to assign a dispatching priority. Code:

```
//stepname EXEC PGM=x,DPRTY=value1
//stepname EXEC PGM=x,DPRTY=(value1,value2)
//stepname EXEC PGM=x,DPRTY=(,value2)
```

### *Examples*

---

```
//S1 EXEC PGM=DEF,DPRTY=(,5)
```

In this example, the parameter is assumed to be DPRTY=(0,5).

---

```
//S2 EXEC PGM=GHI,DPRTY=7
```

---

```
//STEP9 EXEC PROC=JOB6,DPRTY.UP=(,8),DPRTY.DOWN=(4,6)
```

This step calls a cataloged procedure and assigns a dispatching priority to two procedure steps: UP and DOWN. This specification overrides DPRTY parameters on the UP and DOWN EXEC statements, if coded, but does not affect the dispatching priority of any other steps in the procedure.

---

```
//STEP9 EXEC PROC=PROC6,DPRTY=(5,9)
```

This step calls a cataloged procedure and assigns a dispatching priority to all steps in the procedure. This specification overrides all DPRTY parameters in the procedure, if coded.

---



## Performance Control by Performance Group Assignment

Performance groups determine how fast a job executes by controlling the rate at which jobs in the group have access to the processor, the main storage, and the I/O channels. The installation defines the performance groups. Most performance groups designate good processing rates under light system workloads. However, when the system workload is moderate or heavy, some performance groups have much lower processing rates than others.

The installation should define performance groups to meet the response requirements of the jobs to be executed. The installation should maintain a list of these groups.

To associate a job or job step with a performance group, code:

```
//jobname JOB acct,programe,PERFORM=n
//stepname EXEC PGM=x,PERFORM=n
```

*Note:* The PERFORM parameter regulates how a job executes as contrasted with the JES3 `//*MAIN IORATE` parameter, which regulates how a job is scheduled.

For more information on performance, see *SPL: Initialization and Tuning Guide* and *SPL: JES2 Initialization and Tuning* or *SPL: JES3 Initialization and Tuning*.

### Examples

---

```
//J71 JOB , 'ANTHONY B.', PERFORM=52
//STEP1 EXEC PGM=WHIT, PERFORM=4
```

---

## Performance Control by I/O-to-Processing Ratio in a JES3 System

To regulate how a job is scheduled by JES3, code an IORATE parameter:

```
//*MAIN IORATE=xxx
```

The IORATE parameter indicates if the job contains a low, medium, or high number of I/O instructions compared to the number of processing instructions. JES3 uses this value to determine the mix of jobs assigned to a processor: using this parameter, JES3 balances processor-bound processing with I/O-bound processing. A correct balance improves throughput.

### Examples

---

```
//*MAIN IORATE=HIGH
//*MAIN IORATE=LOW
//*MAIN IORATE=MED
```

---



## Part 4. Tasks for Requesting Data Set Resources

This part describes how to create and access data sets. The task required to request a data set is:

- Identification

Other tasks can optionally be performed:

- Description
- Protection
- Allocation
- Processing control
- End processing

### Part 4 Contents

<b>Chapter 12. Identification</b>	12-1
Identification of Data Set	12-2
Permanent Data Set	12-2
Examples	12-2
Members of a Partitioned Data Set	12-2
Examples	12-2
Generations of a Generation Data Group	12-3
Examples	12-3
Areas of an Indexed Sequential Data Set	12-3
Examples	12-3
Temporary Data Set	12-3
Examples	12-4
Members of a Temporary Partitioned Data Set	12-4
Examples	12-4
Areas of a Temporary Indexed Sequential Data Set	12-4
Examples	12-5
Copying the Data Set Name from an Earlier DD Statement	12-5
Example	12-5

## Part 4

Identification of In-Stream Data Set	12-5
Examples	12-6
In-Stream Data Sets in a JES3 System	12-6
Example	12-6
Identification of Data Set on 3540 Diskette Input/Output Unit	12-6
Example	12-7
Identification through Catalog	12-7
Allocation and Deallocation of Catalog Volume	12-7
Using Private Catalogs	12-7
Examples	12-8
Identification through Label	12-8
Label Type for Cataloged or Passed Data Sets	12-9
Nonspecific Volume Request	12-9
Specific Volume Request	12-9
Examples	12-9
Identification by Location on Tape	12-10
Data-Set-Sequence-Number with BLP	12-10
Examples	12-10
Identification as TCAM Message Data Set	12-10
Example	12-10
Identification as Data Set from or to Terminal	12-11
Example	12-11

### Chapter 13. Description 13-1

Description of Status	13-1
Exclusive Control of a Data Set	13-2
Shared Control of a Data Set	13-2
Examples	13-2
Data Set Integrity Processing	13-2
Data Set Integrity Processing for Permanent Data Sets	13-3
Data Set Integrity Processing for Other Data Sets	13-3
Summary of Data Set Integrity Processing	13-4
Description of Data Attributes	13-5
In Data Control Block	13-5
DCB Values from Cataloged Data Sets	13-5
DCB Values from Earlier DD Statements	13-5
Examples	13-5
In Access Method Control Block	13-6
Examples	13-6

### Chapter 14. Protection 14-1

Protection through RACF	14-1
Examples	14-2
Protection for ISO/ANSI/FIPS Version 3 Tapes	14-2
Examples	14-2
Protection by Passwords	14-2
Examples	14-3
Protection of Access to BSAM or BDAM Data Sets	14-3
Other Uses of the LABEL IN Subparameter	14-4
Data Set Processing with LABEL OUT Subparameter	14-4
Examples	14-4

### Chapter 15. Allocation 15-1

Allocation of Device	15-1
----------------------	------

Specifying Device Number	15-2
Specifying Device Type	15-2
Specifying Group Name	15-2
Concurrent Allocation of Devices	15-3
Definition of UNIT Parameters in System Generation	15-3
Requesting More than One Unit	15-3
Number of Devices Allocated	15-4
Volumes Required per DD Statement	15-4
Devices Required per DD Statement	15-4
Devices Assigned per Step	15-4
Relationship of the UNIT and VOLUME Parameters	15-5
Cataloged Data Sets	15-5
Passed Data Sets	15-6
Earlier DD Statement	15-6
Unit and Volume Affinity	15-6
Interaction of Unit and Volume Affinity Requests	15-6
Affinity for Multivolume Data Sets	15-8
Specifying Device for Output Data Set	15-8
Examples	15-9
Device Allocation in a JES3 System	15-11
Device Management	15-11
Device Allocation	15-11
Affect of Job Class on Allocation	15-11
Catalog Use	15-12
Types of JES3 Setup	15-12
Job setup	15-12
High Watermark Setup	15-12
Explicit setup	15-13
Altering JES3 Device Allocation	15-14
Allocation of Volume	15-15
Volume Attributes	15-15
Specific Volume Requests	15-15
How the System Satisfies Specific Volume Requests	15-16
Nonspecific Volume Requests	15-16
How the System Satisfies Nonspecific Volume Requests	15-16
Private Volumes	15-17
Public Volumes	15-17
Volume Affinity	15-18
Explicit Volume Affinity	15-18
Implicit Volume Affinity	15-18
Multivolume Data Sets	15-18
Number of Volumes	15-18
Parallel Mounting	15-18
Processing Order	15-18
Volumes Required per DD Statement	15-18
Mass Storage Volume Groups	15-19
Nonspecific Volume Requests for Mass Storage Volumes	15-19
Specific Volume Requests for Mass Storage Volumes	15-20
Cataloging Data Sets on MSS	15-20
Placing Data Sets on Different MSS Volumes	15-20
Examples	15-20
Allocation of Direct Access Space	15-21
Requesting System Assigned Space	15-21
Requests for Blocks	15-21

## Part 4

Requests for Tracks or Cylinders	15-21
How the System Satisfies the Primary Space Request	15-21
Space on One Volume	15-21
Extents	15-22
System Assigned Space Requests with User Labels	15-22
How the System Satisfies the Secondary Space Request	15-22
Volume for Secondary Space for NEW or MOD Data Set	15-22
Volume for Secondary Space for OLD Data Set	15-22
Secondary Request Only for Current Execution	15-23
Secondary Requests in Blocks	15-23
Directory Space for Partitioned Data Sets	15-23
System Assigned Space Requests for Indexed Sequential Data Sets	15-23
Example	15-23
Requesting Specific Tracks	15-24
Specific Track Requests with User Labels	15-24
Specific Track Requests for Indexed Sequential Data Sets	15-24
Example	15-24
Allocation of Virtual I/O	15-24
Requesting VIO	15-24
Examples	15-25
Backward References to VIO Data Sets	15-25
Examples	15-26
Allocation with Deferred Volume Mounting	15-27
Example	15-27
Allocation with Volume Premounting in a JES2 System	15-27
Example	15-28
Dynamic Allocation	15-28
Example	15-28
<b>Chapter 16. Processing Control</b>	16-1
Processing Control by Suppressing Processing	16-1
Effect of Dummy Data Set	16-1
Requests to Read or Write a Dummy Data Set	16-2
Use of Dummy Data Sets	16-2
Nullifying a Dummy Data Set	16-2
Examples	16-2
Processing Control by Postponing Specification	16-2
How the System Postpones Data Set Definition	16-2
References to the Data Set	16-3
Concatenating DD Statements when DDNAME is Specified	16-3
Use of Postponing Specification	16-3
Examples	16-3
Processing Control with Checkpointing	16-4
Examples	16-4
Processing Control by Subsystem	16-5
Requesting Subsystem	16-5
Example	16-5
Program Control Statements for a Subsystem	16-5
Example	16-5
Processing Control by TCAM Job or Task	16-6
Examples	16-6
<b>Chapter 17. End Processing</b>	17-1
Deallocation End Processing	17-1

Dynamic Deallocation	17-1
Example	17-2
Disposition End Processing of Data Set	17-2
Disposition Controlled by DISP Parameter	17-2
Effect of Abnormal Termination During Execution	17-2
Effect of Abnormal Termination During Allocation	17-2
Effect When No Abnormal Termination Disposition is Coded	17-2
Effect of Device Type on Disposition	17-3
Deleting a Data Set	17-3
Unexpired Expiration Date	17-3
Cataloged Data Sets	17-3
Temporary Data Sets	17-3
Keeping a Data Set	17-3
Cataloging a Data Set	17-4
Use of Cataloging	17-4
CATLG for a Cataloged Data Set	17-4
Generation Data Sets	17-4
When System Does Not Catalog a Data Set	17-4
Uncataloging a Data Set	17-4
Passing a Data Set	17-5
To Pass	17-5
To Receive	17-5
When Passing Step Abnormally Terminates	17-5
Disposition Processing of Unreceived Passed Data Sets	17-5
At Abnormal Termination when Abnormal Termination Disposition is Specified	17-5
At Abnormal Termination when No Abnormal Termination Disposition is Specified	17-6
When Abnormal Termination Occurs Before Execution	17-6
Deletion at End of Job	17-6
Default Disposition Processing	17-6
Bypassing Disposition Processing	17-6
Examples	17-7
Disposition Controlled by Time	17-8
Deleting before Expiration Date or Retention Period	17-8
Examples	17-9
Release of Unused Direct Access Space in End Processing	17-9
Example	17-9
Disposition End Processing of Volume	17-10
RETAIN Support	17-10
Disposition of Removable Volumes	17-10
Tape Volumes in JES2	17-10
Examples	17-10
Volume Retention	17-11
Retained Private Tape Volume	17-11
Retained Public Tape Volume	17-11
Use of Retained Volumes	17-11
Demounting of Passed or Retained Volumes	17-11
Example	17-11





## Chapter 12. Identification

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Identification</b>					
of data set	DSNAME				UPDATE on /*MAIN
of in-stream data set	* or DATA SYSIN DD  DLM		/* or xx delimiter		/*DATASET  /*ENDDATASET
of data set on 3540 Diskette Input/Out- put Unit	DSID				
through catalog	JOBCAT DD STEP CAT DD				
through label	label-type on LABEL				
by location on tape	data-set- sequence-number on LABEL				
as TCAM message data set	QNAME				
from or to terminal	TERM				

Figure 12-1. Identification Task for Requesting Data Set Resources

## Identification

### Identification of Data Set

When creating a data set, assign a name to the data set in the DSNNAME parameter. The data set name is stored with the data set. When a later step or job uses the data set, identify the data set in the DSNNAME parameter; the system uses the data set name to locate the data set on the volume.

How you code the DSNNAME parameter depends on the type of data set and whether it is permanent or temporary or it is copied from an earlier DD statement.

#### Permanent Data Set

Identify a permanent data set by coding:

<b>DSNNAME = dsname</b>	For a permanent data set
<b>DSNNAME = dsname(member)</b>	For a member of a permanent partitioned data set
<b>DSNNAME = dsname(generation)</b>	For a generation of a permanent generation data group
<b>DSNNAME = dsname(area)</b>	For an area of a permanent indexed sequential data set

To create a permanent data set, assign it a name in the DSNNAME parameter and a disposition of KEEP or CATLG in the DISP parameter. The DISP subparameter makes it a permanent data set. To use the data set, specify the data set's name in the DSNNAME parameter in a later step or job or a backward reference to the creating DD statement in a later step in the same job.

#### *Examples*

---

```
//MYDS DD DSNNAME=PLANA,DISP=(NEW,KEEP,DELETE),
//      UNIT=3380,VOLUME=SER=167833,SPACE=(CYL,(10,5))

//DSC DD DSNNAME=PLANB,DISP=(NEW,CATLG,DELETE),
//      UNIT=3350,VOLUME=SER=275566,SPACE=(TRK,(20,5))

//OLDDS DD DSNNAME=EXIST,DISP=OLD
```

---

#### Members of a Partitioned Data Set

A partitioned data set consists of sequential records in independent groups, which are called members: each member is identified by a member name in a directory. To add a member to a partitioned data set or retrieve a member, specify the partitioned data set name followed by the member name in parentheses.

#### *Examples*

---

```
//NEWA DD DSNNAME=RPRT(WEEK1),DISP=(NEW,CATLG,DELETE),
//      UNIT=3380,VOLUME=SER=236688,SPACE=(CYL,(20,5,20))

//ADD1 DD DSNNAME=RPRT(WEEK2),DISP=OLD
```

---

## Generations of a Generation Data Group

A generation data group is a collection of chronologically related data sets that have the same data set name. To add a generation to a generation data group or retrieve a generation, specify the generation data group name followed by the generation number. A zero is the current generation of the group, a negative number (for example, -1) is an older generation, a positive number (for example, +1) is a new generation that is not yet cataloged.

### Examples

---

```
//NEWGDS DD DSNAME=GDS(0),DISP=(NEW,CATLG,DELETE),
//        UNIT=3380,VOLUME=SER=334455,SPACE=(CYL,20)

//OLDGDS DD DSNAME=GDS(-1),DISP=OLD

//NEWER  DD DSNAME=GDS(+1),DISP=(NEW,CATLG,DELETE),
//        UNIT=3350,VOLUME=SER=222333,SPACE=(TRK,15)

//ALLG   DD DSNAME=GDS,DISP=OLD
```

---

## Areas of an Indexed Sequential Data Set

An indexed sequential data set consists of three areas: index, prime, and overflow. To create the data set, define each area by identifying the data set name followed by the area name. The area name is PRIME, INDEX, or OVFLOW. To define the data set on one DD statement, code DSNAME=dsname or DSNAME=dsname(PRIME). To retrieve the data set, code only the data set name.

### Examples

---

```
//NEWIS DD DSNAME=ISDS(PRIME),DISP=(NEW,CATLG,DELETE),
//        UNIT=3350,VOLUME=SER=222333,SPACE=(TRK,15)
//        DD DSNAME=ISDS(INDEX),DISP=(NEW,CATLG,DELETE),
//        UNIT=3350,VOLUME=SER=222333,SPACE=(TRK,5)
//        DD DSNAME=ISDS(OVFLOW),DISP=(NEW,CATLG,DELETE),
//        UNIT=3350,VOLUME=SER=222333,SPACE=(TRK,10)

//OLDIS DD DSNAME=ISDS,DISP=OLD
```

---

## Temporary Data Set

Any data set that is created and deleted in the same job is a temporary data set. Identify a temporary data set by coding:

<b>DSNAME = &amp;&amp;dsname</b>	For a temporary data set
<b>DSNAME = &amp;&amp;dsname(member)</b>	For a member of a temporary partitioned data set
<b>DSNAME = &amp;&amp;dsname(area)</b>	For an area of a temporary indexed sequential data set
<b>No DSNAME parameter</b>	For a temporary data set to be named by the system

The system generates a qualified name for the temporary data set. The name begins with SYS and includes the Julian date, the time, the job name, the temporary name assigned in the DSNAME parameter, if specified, or an identifying name and number, if a DSNAME is not specified, and other identifying characters.

## Identification

The time in the system-generated qualified name is the same for all temporary data sets in a job. Therefore, if the same temporary data set name appears more than once in a job, the system may create duplicate data set names, which would be a JCL error.

If the DISP parameter for a temporary data set specifies KEEP or CATLG, the system changes the disposition to PASS and deletes the data set at job termination. However, the system does not change the disposition for a data set when all of the following are true:

- The data set resides on tape
- The data set is new
- The data set is not named in a DSNAME parameter
- The status in the DISP parameter is OLD or SHR
- The UNIT parameter contains DEFER

In this case, the system deletes the data set at job termination but tells the operator to keep the volume for the data set.

### *Examples*

---

```
//TEMPDS1 DD DSNAME=&&MYDS,DISP=NEW,UNIT=3350,  
//        SPACE=(CYL,20)  
  
//TEMPDS2 DD DSNAME=&&DSA,DISP=(NEW,PASS),UNIT=3380,  
//        SPACE=(TRK,15)
```

---

### Members of a Temporary Partitioned Data Set

To add a member to a temporary partitioned data set or retrieve a member during the job, specify the partitioned data set's temporary name and follow it with the member name in parentheses.

### *Examples*

---

```
//TEMPMEM DD DSNAME=&&DS1(MEM1),DISP=(NEW,PASS),  
//        UNIT=3380,SPACE=(CYL,(20,,2))  
  
//GETMEM DD DSNAME=&&DS1(MEM1),DISP=OLD
```

---

### Areas of a Temporary Indexed Sequential Data Set

To create a temporary indexed sequential data set and define any of its areas on a DD statement, identify the data set's temporary name followed by the area name. To define the temporary data set on one DD statement, code DSNAME = &&dsname or DSNAME = &&dsname(PRIME). To retrieve the temporary data set in the same job, code DSNAME = &&dsname.

## Examples

---

```
//TEMPIS DD DSNAME=&&ISDS(PRIME),DISP=(NEW,PASS),
// UNIT=3380,SPACE=(CYL,20)
// DD DSNAME=&&ISDS(INDEX),DISP=(NEW,PASS),
// UNIT=3380,SPACE=(CYL,5)
// DD DSNAME=&&ISDS(OVFLOW),DISP=(NEW,PASS),
// UNIT=3380,SPACE=(CYL,10)

//ANOTHER DD DSNAME=&&ISDS2,DISP=(NEW,PASS),UNIT=3350,
// SPACE=(TRK,50)

//OLDIS DD DSNAME=&&ISDS2,DISP=OLD
```

---

## Copying the Data Set Name from an Earlier DD Statement

If a data set name is used several times in a job, copy it from the DD statement that uses it first. It can be copied whether it is specified in the DSNAME parameter or assigned by the system. Use copying to make changing data sets from job to job easier and to eliminate having to assign names to temporary data sets. Copy a data set name by coding:

```
//ddname DD DSNAME=*.ddname
//ddname DD DSNAME=*.stepname.ddname
//ddname DD DSNAME=*.stepname.procstepname.ddname
```

## Example

---

```
//COPYDS DD DSNAME=*.MYDS
```

---

## Identification of In-Stream Data Set

Enter data through the input stream by coding one of the following:

```
//ddname DD *
//ddname DD DATA
```

A step can contain more than one in-stream data set. Use the DD DATA statement when the data contains JCL statements.

If the statement that begins the data set contains a DLM parameter, end the in-stream data set with a statement containing the two characters in the DLM parameter. Otherwise, end the in-stream data set with either of the following delimiters:

```
/*
Another JCL statement, if begun with a DD * statement
```

## Identification

### *Examples*

---

```
//DSIN DD *
      .
      (data)
      .
//INSET DD DATA
      .
      (data)
      .
/*
//THIRD DD *,DLM=ED
      .
      (data)
      .
ED
```

---

## In-Stream Data Sets in a JES3 System

In a JES3 system, an in-stream data set can also begin with a `/*DATASET` statement and end with a `/*ENDDATASET` statement. The `/*DATASET` statement must start an in-stream data set that is used as input to a dynamic support program (DSP).

### *Example*

---

```
//J1 JOB 2233,'K.A. BRAND'
//S1 EXEC PGM=MYPROG
/*DATASET DDNAME=S1.MYDD4,J=YES
      .
      data
      .
/*ENDDATASET
```

---

## Identification of Data Set on 3540 Diskette Input/Output Unit

IBM 3540 diskette volumes can contain associated data sets. Associated data sets are treated like in-stream data sets and are spooled in as SYSIN data sets. These associated data sets are identified by coding a DSID parameter and, optionally, a volume serial on a DD \* or DD DATA statement in the input stream:

```
//ddname DD *,DSID=xxxx,VOLUME=SER=yyyyyy
```

To merge associated data sets into the job input stream, the stream containing the DD statements for the associated data sets must be processed by the diskette reader program, rather than by JES2 or JES3.

For more information on the 3540 diskette, see *IBM 3540 Programmer's Reference*..

## Example

---

```
//ASSTDS DD DATA,DSID=3254,VOLUME=SER=778356
```

---

## Identification through Catalog

A system or private catalog contains pointers to existing, cataloged data sets. The system uses these pointers to locate data sets when a DD statement requests an old data set without UNIT or VOLUME parameters. For example:

```
//ddname DD DSNAME=dsname,DISP=OLD
```

**Allocation and Deallocation of Catalog Volume:** When the DSNAME parameter requests a cataloged data set, the system mounts the catalog volume, if it is not already mounted. From the catalog, the system obtains the pointer to the requested data set. Later, if the device on which the catalog is mounted is needed for another volume, the system demounts the catalog volume. The system assigns the catalog to the job step and performs disposition processing for the catalog volume when the job step ends.

In the following cases, the system does not mount the catalog volume during disposition processing of a job's data sets:

- The job abnormally terminates and data sets with an abnormal termination disposition of CATLG or UNCATLG were passed by a job step but not received by a later step.
- The system deallocates a step's data sets during warm start.

### Using Private Catalogs

Private catalogs are defined on JOBCAT DD or STEPCAT DD statements. To define a private catalog, use access method services, as explained in *VSAM Administration Guide*. The system searches a private catalog before a system catalog when a JOBCAT or STEPCAT DD statement appears in the job or step and a DD statement does not specify unit and volume serial information for a data set. A JOBCAT catalog applies to each step of a job in which a STEPCAT catalog is not specified.

*Note:* In a JES3 system, a private catalog must be on a permanently resident volume.

To locate a data set, the system searches catalogs in the following order:

1. Private catalog(s) specified in the current step in a STEPCAT DD statement and statements concatenated to it.
2. If no private catalogs are specified for the job step, private catalogs specified in the current job in a JOBCAT DD statement and statements concatenated to it.
3. A CVOL or private catalog indicated by the first qualifier, if any, of the data set name.
4. The master catalog.

# Identification

## Examples

---

```
//CATDS DD DSNAME=DS1,DISP=OLD
//ANOTH DD DSNAME=A.B.C,DISP=OLD
//JOB CAT DD DSNAME=PRIVCAT1,DISP=SHR
// DD DSNAME=CONCAT2,DISP=SHR
//STEP CAT DD DSNAME=PRIVCATS,DISP=SHR
```

---

## Identification through Label

The system uses data set labels to:

- Identify volumes and the data sets they contain.
- Store data set attributes.

A label is either standard or nonstandard. Standard labels can be processed by the system; nonstandard labels must be processed by user-written routines, which the installation adds to the system.

Data sets on tape volumes usually have labels; these labels can be standard or nonstandard. If labels are present, they precede each data set on the volume. Data sets on direct access volumes always have labels; these labels must be standard. Direct access labels are in the volume table of contents (VTOC) for the volume.

The label type subparameter tells the system the type of labels for the data set. The label type is coded:

```
//ddname DD LABEL=(,label)...
```

The label types are:

**SL: IBM standard labels**

**SUL: both IBM standard and user labels**

For data sets on direct access, only SL or SUL can be specified. For SL or SUL, or when the label type subparameter is omitted because the data set has IBM standard labels, the system ensures that the correct tape or direct access volume is mounted.

**AL: ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels**

**AUL: ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels, and ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 user labels**

For AL or AUL, the system ensures that the correct tape volume is mounted; the tape must have an ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 label.

**NSL: nonstandard labels**

For NSL, installation-provided nonstandard label processing routines must ensure that the correct tape volume is mounted.



**NL: no labels**

**BLP: bypasses label processing**

For NL or BLP, the operator must ensure that the correct tape volume is mounted. If you specify NL, the data set must not have any standard labels.

*Use of BLP:* BLP is not a label type, but a request that the system bypass label processing. Use this specification for a blank tape or for overwriting a seven-track tape of a parity or density different than its current parity or density.

**LTM: bypasses a leading tape mark on unlabeled tape**

**Label Type for Cataloged or Passed Data Sets:** For cataloged and passed data sets, the system does not keep label type information. Therefore, when referring to a cataloged or passed data set that has other than standard labels, code the LABEL type subparameter.

**Nonspecific Volume Request:** The label type subparameter can be specified for a nonspecific tape volume request, that is, a DD statement with no volume serial numbers. If the operator mounts a tape volume with a different label type, the system requests that the operator mount another volume. But, if the specified label type is NL or NSL for the nonspecific volume request and the operator mounts a volume with standard labels, the system uses the volume if **both** of the following are true:

1. The expiration date of the existing data set on the volume is passed.
2. The existing data set on the volume is not password protected.

If you specify SL on 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 writing the IBM standard labels. 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.

**Specific Volume Request:** If you specify SL on a specific volume request, that is, a DD statement that specifies volume serial numbers, but the volume does not contain IBM standard labels:

- If the mounted volume contains labels, the system rejects the volume and asks the operator to mount the specified tape volume.
- If the mounted volume is not labeled, the system asks the operator whether to reject the volume or write standard labels on it.

## Examples

---

```
//DSF DD DSN=ALLAB,LABEL=(,AL),UNIT=3420,  
// VOLUME=SER=223344,DISP=(NEW,CATLG)  
  
//DSJ DD DSN=CATDS,DISP=OLD,LABEL=(,SUL)
```

---

## Identification

### Identification by Location on Tape

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 second, third, fourth, etc., on the volume. Code the data set sequence number to position the tape:

```
//ddname DD LABEL=(data-set-sequence-number,label),...  
//ddname DD LABEL=data-set-sequence-number,...
```

**Data-Set-Sequence-Number with BLP:** If you specify BLP for the label type, the system treats anything between tapemarks as a data set. Therefore, if the tape actually has labels, code the data-set-sequence-number subparameter to position the tape properly; the subparameter must reflect all labels and data sets that precede the desired data set. The *Magnetic Tape Labels and File Structure Administration* publication illustrates where tapemarks appear.

#### Examples

---

```
//DDEX1 DD DSNAME=TAPEDS3,DISP=(NEW,KEEP),UNIT=3420,  
// LABEL=(3,SL),VOLUME=SER=666555  
  
//DDEX2 DD DSNAME=TAPEDS4,DISP=(NEW,KEEP),UNIT=3420,  
// LABEL=(8,BLP),VOLUME=SER=223344
```

---

### Identification as TCAM Message Data Set

To identify a data set as containing telecommunications access method (TCAM) messages, code the following:

```
//ddname DD QNAME=procname  
//ddname DD QNAME=procname.tcamname
```

The QNAME parameter refers to a TPROCESS macro instruction that defines a destination queue for the messages. The parameter can also name a TCAM job to process the messages.

#### Example

---

```
//EX1 DD QNAME=MACRO1.TJOB
```

---

## Identification as Data Set from or to Terminal

In a job run in a time sharing option (TSO) system, identify a data set as coming from or going to the terminal in the JOB statement USER parameter by coding:

```
//ddname DD TERM=TS
```

In a background or batch job, the system treats the TERM=TS parameter as a SYSOUT=\* parameter if no other parameters are coded.

### *Example*

---

```
//MYTSODS DD TERM=TS
```

---



## Chapter 13. Description

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Description					
of status	DISP				
of data attributes	DCB AMP				

Figure 13-1. Description Task for Requesting Data Set Resources

## Description of Status

The process of securing control of data sets for 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 data set FILE.

- READ wants only to read and copy certain records
- MODIFY deletes some records and changes other records

If both jobs use FILE concurrently, READ cannot be certain of the integrity of FILE because MODIFY is changing records in the data set. MODIFY should have exclusive control of the data set.

Indicate the type of control needed by coding the data set's status:

```
//ddname DD DISP=(NEW,...
//ddname DD DISP=(OLD,...
//ddname DD DISP=(MOD,...
//ddname DD DISP=(SHR,...
```

For exclusive use of a data set, code:

- NEW: the data set is being created in this job step.
- OLD: the data set existed before this job step.

## Description

- **MOD:** the system first assumes that the data set exists. For an existing sequential data set, MOD 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.

If the system cannot find volume information for the data set in the catalog or passed with the data set from a previous step, the system assumes that the data set is being created in this job step. For a new data set, MOD causes the read/write mechanism to be positioned at the beginning of the data set.

For shared use of a data set, code:

- **SHR:** the data set existed before this job step and can be read by other concurrent jobs.

**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 completion of the last step in the job. 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. See *Data Administration: Macro Instruction Reference*.

**Shared Control of a Data Set:** Several jobs can concurrently use a data set on a direct access device if they request shared control of the data set. None of the jobs should change the data set in any way.

If more than one step requests a shared data set, code SHR on every DD statement that requests the data set, if it is to be used by concurrently executing jobs.

### Examples

---

```
//DD1 DD DSANME=PERMDS,DISP=OLD
//DD2 DD DSNNAME=&&TEMPDS,DISP=NEW
//DD3 DD DSNNAME=GENDS(+1),DISP=(NEW,CATLG)
```

---

## Data Set Integrity Processing

The system performs data set integrity processing once for each job, for the following types of data sets:

- Permanent data sets
- Non-virtual I/O (VIO) temporary data sets
- Data sets with alias names, created with the access method services DEFINE command; see *Integrated Catalog Administration: Access Method Services Reference* or *VSAM Catalog Administration: Access Method Services Reference*
- Members of generation data groups

The system **does not** perform data set integrity processing for subsystem data sets.

**Data Set Integrity Processing for Permanent Data Sets:** To secure control for all permanent data sets for the job, the system enqueues each data set, marking the data set as requested by that job and noting the kind of control requested: shared or exclusive. The system assigns control of the data set until the last step in the job completes.

If you code NEW, OLD, or MOD on any DD statement for a data set, the system assigns exclusive control. A statement requesting exclusive control overrides any number of statements requesting shared control.

The job receives control of the data set if:

- Another job is not using the data set.
- 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 no exclusive requests are 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.
- Another job is using the data set, with either exclusive or shared control, and this job requests exclusive control.
- Another job is using the data set, with shared control, and yet another, earlier job requests exclusive control.

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 is starting the job waits until the required data sets become available, unless the operator cancels the job.

When the system has secured control of all permanent 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.

**Data Set Integrity Processing for Other Data Sets:** 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 way as for permanent 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 and if the job cannot secure control of 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.

# Description

## Summary of Data Set Integrity Processing

	Data set is currently in use:		Data set is not in use	Data set is previously requested for:	
	Shared control	Exclusive control		Shared control	Exclusive control
<b>Permanent data set requested for:</b>					
<b>Shared control</b>	Request granted	Request granted when data set released	Request granted	Request granted	Request granted when data set released
<b>Exclusive control</b>	Request granted when data set released	Request granted when data set released	Request granted	Request granted when data set released	Request granted when data set released
<b>Non-VIO temporary data set requested for:</b>					
<b>Shared control</b>	Request granted	Request not granted; requesting job terminated	Request granted	Request granted	Request not granted; requesting job terminated
<b>Exclusive control</b>	Request not granted; requesting job terminated	Request not granted; requesting job terminated	Request granted	Request not granted; requesting job terminated	Request not granted; requesting job terminated
<b>GDG data set requested for:</b>					
<b>Shared control</b>	Request granted	Request not granted; requesting job terminated	Request granted	Request not granted; requesting job terminated	Request granted
<b>Exclusive control</b>	Request not granted; requesting job terminated	Request not granted; requesting job terminated	Request granted	Request not granted; requesting job terminated	Request not granted; requesting job terminated
<b>Data set with alias name requested for:</b>					
<b>Shared control</b>	Request granted	Request not granted; requesting job terminated	Request granted	Request granted	Request not granted; requesting job terminated
<b>Exclusive control</b>	Request not granted; requesting job terminated	Request not granted; requesting job terminated	Request granted	Request not granted; requesting job terminated	Request not granted; requesting job terminated

Figure 13-2. Data Set Integrity Processing



## Description of Data Attributes

The system obtains information needed to read from and write to a data set from the data control block (DCB) or, for a VSAM data set, from the access method control block (ACB).

### In Data Control Block

The system obtains data control block information from the following sources, in override order:

- The DCB macro instruction, in assembler language programs, or file definition statements or language-defined defaults in programs in other languages.
- The DCB subparameters on the DD statement.

```
//ddname DD DCB=subparameter,...
//ddname DD DCB=(subparameter,subparameter,...),...
```

- The data set label.

Therefore, the system ignores a value in a DCB subparameter on the DD statement if the data control block already contains the value. The system ignores a value in the data set label if the data control block already contains the value from the program or a DD DCB subparameter.

**DCB Values from Cataloged Data Sets:** The DD statement DCB parameter can ask the system to copy certain values from the data set label of a cataloged data set, by coding:

```
//ddname DD DCB=dsname,...
//ddname DD DCB=(dsname,subparameter,...)...
```

The system copies the DSORG, RECFM, OPTCD, BLKSIZE, LRECL, KEYLEN, and RKP values from the label. If any of these values are coded in subparameters following the dsname, the system uses the coded values.

**DCB Values from Earlier DD Statements:** The DD statement DCB parameter can ask the system to copy all subparameters from the DCB parameter in an earlier DD statement, by coding a backward reference to the earlier statement:

```
//ddname DD DCB=*.ddname
//ddname DD DCB=*.stepname.ddname
//ddname DD DCB=*.stepname.procstepname.ddname
```

### Examples

---

```
//S1 EXEC PGM=ANYA
//DD1 DD DSNAME=ABC,DCB=(RECFM=FB,LRECL=80,BLKSIZE=960),
// DISP=(NEW,CATLG,DELETE),UNIT=3380,VOLUME=223344,
// SPACE=(CYL,(30,10))
//S2 EXEC PGM=ANYB
//DD2 DD DSNAME=COPIER1,DCB=ABC
//S3 EXEC PGM=ANYC
//DD3 DD DSNAME=COPIER2,DCB=*.S1.DD1
```

---

# Description

## In Access Method Control Block

The system obtains access method control block information from the following sources, in override order:

- The AMP subparameters on the DD statement.

```
//ddname DD AMP=(subparameter),...  
//ddname DD AMP=('subparameter,subparameter,...'),...
```

- The ACB, EXLST, or GENCB macro instructions in assembler language programs.
- The catalog entry for the data set.

Therefore, the system ignores a value in a program macro instruction if the DD AMP parameter supplies the value. The system ignores a value in the data set catalog entry if the access method control block already contains the value from a DD AMP subparameter or a macro instruction in the program.

*Note:* The override order for ACB values is different from the override order for DCB values.

### *Examples*

---

```
//DD4 DD DSN=ANYVSAM1,AMP=('BUFND=4,BUFNI=4,STRNO=2'),  
// DISP=(NEW,CATLG,DELETE),UNIT=3380,VOLUME=556677,  
// SPACE=(TRK,(200,50))
```

---

## Chapter 14. Protection

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Protection</b>					
through RACF	PROTECT				
for ISO/ANSI/FIPS Version 3 tapes	ACCODE				
by passwords	PASSWORD and NOPWREAD on LABEL				
of access to BSAM and BDAM data sets	IN and OUT on LABEL				

Figure 14-1. Protection Task for Requesting Data Set Resources

### Protection through RACF

To ask for Resource Access Control Facility (RACF) protection, code:

```
//ddname DD PROTECT=YES,...
```

Through the PROTECT parameter, RACF Version 1 Release 6 and earlier can protect the following:

- A data set on a direct access volume.
- A tape volume with standard labels, that is:

```
LABEL=(,SL)
LABEL=(,SUL)
LABEL=(,AL)
LABEL=(,AUL)
```

For more information, see *Resource Access Control Facility (RACF) Security Administrator's Guide*.

## Protection

### Examples

---

```
//TAPE1 DD DSNAME=EXIST1,PROTECT=YES,DISP=OLD,  
//      VOLUME=(, , ,1,2,SER=(223344,556677)),  
//      UNIT=(3400-5,2),LABEL=(,SUL)  
  
//DISKDS DD DSNAME=NEWS2,PROTECT=YES,DISP=(NEW,CATLG,KEEP),  
//      VOLUME=223344,UNIT=3380
```

---

## Protection for ISO/ANSI/FIPS Version 3 Tapes

To control access to an ISO/ANSI/FIPS Version 3 tape data set, code:

```
//ddname DD ACCODE=access-code,...
```

The system must contain an installation-written file-access exit routine. This routine verifies that the ACCODE parameter specifies the correct code for an existing data set and, therefore, can use a data set.

### Examples

---

```
//DD1 DD DSNAME=NEWS,ACCODE=F,LABEL=(,AL),UNIT=3380,  
//     VOLUME=998877,DISP=(NEW,CATLG,KEEP)  
  
//DD2 DD DSNAME=OLDDS,ACCODE=J,LABEL=(,AL),UNIT=3380,  
//     VOLUME=665544,DISP=OLD
```

---

## Protection by Passwords

To protect a data set with a password, code:

```
//ddname DD LABEL=(data-set-sequence-number,label,PASSWORD)  
//ddname DD LABEL=(data-set-sequence-number,,PASSWORD)  
//ddname DD LABEL=(, ,PASSWORD)
```

To use a password-protected data set, code:

```
//ddname DD LABEL=(data-set-sequence-number,label,PASSWORD)  
//ddname DD LABEL=(data-set-sequence-number,,PASSWORD)  
//ddname DD LABEL=(, ,PASSWORD)  
//ddname DD LABEL=(data-set-sequence-number,label,NOPWREAD)
```

These subparameters mean the following:

- **PASSWORD:** The data set cannot be read from, written to, or deleted by another job or step unless the operator supplies the system with the correct password.

- **NOPWREAD:** The data set cannot be written to or deleted by another job or step unless the operator supplies the system with the correct password. However, the data set can be read without the password.

To protect a data set with a password, 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.

### Examples

---

```
//EX1 DD DSNAME=ABC,DISP=(NEW,CATLG,DELETE),
//      LABEL=(,SL,PASSWORD),UNIT=3400-5,VOLUME=223344

//EX2 DD DSANME=DEF,DISP=OLD,LABEL=(,SL,NOPWREAD)
```

---

## Protection of Access to BSAM or BDAM Data Sets

The **LABEL** parameter can modify the data set processing through the **IN** and **OUT** subparameters, as indicated in Figure 14-2, if the assembler **OPEN** macro instruction specifies the data set processing as:

- When using the basic sequential access method (BSAM): **INOUT**, **OUTIN**, **OUTINX**, or **EXTEND**
- When using the basic direct access method (BDAM): **UPDAT**

The **LABEL** subparameters are coded:

```
//ddname DD LABEL=(data-set-sequence-number,label,PASSWORD,IN)
//ddname DD LABEL=(,label,PASSWORD,OUT)
//ddname DD LABEL=(,,NOPWREAD,IN)
//ddname DD LABEL=(,,,OUT)
```

OPEN Macro Parameter	LABEL Subparameter	Program Processing of Data Set	Required Password
INOUT (BSAM) UPDAT (BDAM)	IN	Read records (If the program tries to write to the data set, the system gives control to the error analysis (SYNAD) routine.)	Read password, if data set protected with <b>PASSWORD</b> ; write password, if data set protected with <b>NOPWREAD</b>
OUTIN (BSAM) UPDAT (BDAM)	OUT	Write records (If the program tries to read the data set, the system gives control to the error analysis (SYNAD) routine.)	Write password, if data set protected with <b>PASSWORD</b> or <b>NOPWREAD</b>
OUTINX (BSAM) EXTEND (BSAM)	OUT	Add records to end of data set (If the program tries to read the data set, the system gives control to the error analysis (SYNAD) routine.)	Write password, if data set protected with <b>PASSWORD</b> or <b>NOPWREAD</b>

Figure 14-2. Processing with DD LABEL Subparameter IN or OUT

## Protection

**Other Uses of the LABEL IN Subparameter:** You can also use the IN subparameter to avoid operator intervention when reading a data set that has an unexpired expiration date.

**Data Set Processing with LABEL OUT Subparameter:** When the OPEN macro instruction specifies OUTINX or EXTEND and the DD LABEL contains an OUT subparameter, the system adds records to the end of the data set regardless of the DISP parameter of the DD statement.

### Examples

---

```
//EX1 DD DSNAME=D.E.F,DISP=OLD,LABEL=(, ,NOPWREAD,IN)  
//EX2 DD DSNAME=EXIST,DISP=MOD,LABEL=(, ,PASSWORD,OUT)
```

---

## Chapter 15. Allocation

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Allocation</b>					
of device	UNIT		CLASS on JOB (JES3 only)		SETUP, MSS, and CLASS on /*MAIN
of tape or direct access volume	VOLUME MSVGP				EXPDTCHK and RINGCHK on /*MAIN
of direct access space	SPACE				
of virtual I/O	UNIT DSNAME = tem- porary data set				
with deferred vol- ume mounting	DEFER on UNIT				
with volume pre- mounting				/*SETUP	
dynamic			DYNAMNBR on EXEC		

Figure 15-1. Allocation Task for Requesting Data Set Resources

### Allocation of Device

On the DD statement for a data set, indicate the device on which the data set resides or is to be written by coding a UNIT parameter. The UNIT parameter can specify:

- A particular device:

```
//ddname DD UNIT=device-number,...
```

- A type of device, for example, a 3350 direct access device or a 1403 printer:

```
//ddname DD UNIT=device-type,...
```

## Allocation

- A group of devices, for example, DISK to indicate all direct access devices in the system:

```
//ddname DD UNIT=group-name,...
```

The status of a device affects whether the system can allocate it or not. See Figure 15-2.

Status	Device Type				
	Direct Access	Tape	Printer Punch	Graphic	Teleprocessing
Online	Eligible for allocation				
Offline	Eligible for allocation when the operator brings the device online				Eligible for allocation
Pending Unload	Eligible for allocation when the 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 15-2. Affect of Device Status on Allocation

**Specifying Device Number:** The **device number** is a 3-character identifier assigned to the device when it is installed. For example, UNIT=180 can identify a particular storage device. Specifying a device number limits assignment: the system can assign only that specific device. If the device is already being used, the job must be delayed or canceled. Specify a device number only when necessary.

**Specifying Device Type:** Requesting a **device type** allows the system to assign any available device of that type. For example, UNIT=3350 indicates that you want the system to assign any available 3350 Direct Access Storage device. For more information on specifying device types, see *Installation: System Generation*.

**Specifying Group Name:** During system generation, the installation can define **group names** for a group of devices. The devices in a group may or may not all be the same type. Requesting a group name allows the system to assign any available device in the group. For example, if the group named DISK includes 3350 and 3380 Direct Access Storage devices, the system assigns an available 3350 or 3380 device when UNIT=DISK is coded. If the group named 3350A includes three particular 3350 devices, the system assigns one of these 3350 devices when UNIT=3350A is coded.

**Groups with Several Types of Devices:** If the group contains more than one type of device and the DD statement requests more than one device, the system allocates devices of the same type from the group. For example, if the group named TAPE includes both 3400-5 and 3400-6 devices and the DD statement specifies UNIT=(TAPE,2), the system assigns either two 3400-5s or two 3400-6s. If the system does not have enough devices of one type to satisfy the request, the system terminates the job.



If a group contains more than one type of device, do not code the group name when requesting an existing data set or a specific volume. The system may assign one type of device while the data set resides on another type. For example, if SYSSQ contains all tape and direct access devices, do not code UNIT=SYSSQ for an existing data set on tape; the system might assign a direct access device.

*Groups with Devices with Special Features:* This rule also applies 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 Disk/Storage - Fixed Head Feature User's Guide*.

If a nonspecific volume request requires more than one tape device 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.

*Concurrent Allocation of Devices:* Only direct access devices can be allocated to different jobs executing concurrently. Teleprocessing equipment cannot be allocated more than once in the same job step. If a printer, punch, teleprocessing equipment, or graphics device is designated as a console, it cannot be allocated to a job.

*Definition of UNIT Parameters in System Generation:* The installation describes each device to the system during system generation. During this process, the installation defines the device types and group names to be coded in the DD UNIT parameter.

The installation should maintain a list of the device types and group names. For more information, see *Installation: System Generation*.

## Requesting More than One Unit

For faster processing, request several units for a multivolume data set or for a data set that may require additional volumes. When each volume is on its own device, step execution is not halted while the operator demounts and mounts volumes.

Always request several units when the data set resides on more than one permanently resident or reserved volumes or may be extended to a new volume during step execution. Permanently resident and reserved volumes cannot be demounted in order to mount a new volume.

Request multiple units by:

- Coding the unit count subparameter:

```
//ddname DD UNIT=(device,unit-count),...
```

- Requesting parallel mounting when the VOLUME parameter requests more than one volume in the volume count parameter or in more than one serial number:

```
//ddname DD UNIT=(device,P),VOLUME=(,,,volume-count)
//ddname DD UNIT=(device,P),
//          VOLUME=SER=(serial-number,serial-number,...)
```

# Allocation

## Number of Devices Allocated

The system assigns volumes and devices for a job step by calculating the following:

- The maximum number of volumes per DD statement
- The maximum number of devices per DD statement
- The number of devices for the step

**Volumes Required per DD Statement:** See “Volumes Required per DD Statement” on page 15-18.

**Devices Required per DD Statement:** The maximum number of tape devices or direct access devices required to satisfy any DD statement is the unit count in the UNIT parameter. However, if the UNIT parameter also specifies P, for parallel mount, the system uses the **greatest** of the following numbers to determine how many devices and volumes to allocate:

- unit-count in the UNIT parameter
- volume-count specified in the VOLUME parameter
- number of serial numbers implicitly or explicitly specified

The number of devices is affected by the DD statement parameters as follows:

DD Statement Specifies	System Action
UNIT = AFF	The system obtains the device requirements from the referenced DD statement. All of the devices used for the referenced DD statement are shared with the referring statement's data set.
Generation data group (GDG)	The system determines the number of devices needed by totaling the devices needed for each generation data set. Each generation data set is handled as a single request.
VSAM data set	The system determines the number of devices needed based on the device/volume configuration of the data set. If the data set is on more than one type of device, the system determines the total number of devices required and allocates them. The system may override the unit count or parallel mounting, if specified.
Unit name that includes different device types	The system allocates devices of the same type.

**Devices Assigned per Step:** The number of devices assigned for a job step is not necessarily the sum of the device requirements for each DD statement.

The following tend to reduce the total devices assigned for a step:

- A volume can be allocated to only one device. Therefore, when more than one DD statement asks for the same volume, the system allocates the same volume on the same device.
- Requests for direct access space on public and/or storage volumes can be allocated to the same volume. Therefore, when more than one DD statement requests such space, the system can allocate the same volume on the same device.
- Requests for the same public tape volume are allocated to that volume. Therefore, if a DD statement requests a public tape and specifies VOLUME = REF, the system can allocate the same volume on the same device.

The following tend to increase the total devices assigned for a step:

- A permanently resident or reserved volume cannot be demounted. Therefore, the system assigns a permanently resident or reserved volume to its own device, on which it is mounted. The volume is assigned to its own device even if the DD statements specify that the device was to be shared with other volumes.
- A direct access volume is requested by more than one DD statement in a step; the volume is shared by the data sets. The system assigns that volume to a device and does not assign any other volumes to that device, even if the DD statements specify that the device was to be used for other volumes.
- The system allocates additional devices for a VSAM data set, if the data set resides on more than one type of device.
- The system allocates a direct access device for a private catalog, if it is associated with and/or used to retrieve volume information about a requested data set.
- For a generation data group (GDG) on direct access, the system may have to assign additional devices to satisfy the device type needs for each generation data set in the GDG.
- When DD statements request conflicting device assignments for a tape volume, the system assigns the volume involved in the conflict its own device. For example:

```
//DD1 DD UNIT=2400,VOLUME=SER=(V1,V2)
//DD2 DD UNIT=2400,VOLUME=SER=(V2,V3)
```

Volume serial V2 has conflicting device assignments. Therefore, the system assigns the three volumes to three devices. If the DD2 had requested unit affinity, UNIT=AFF=DD1, the system would have assigned only one device to all three volumes.

### Relationship of the UNIT and VOLUME Parameters

The system can obtain unit information from sources other than the UNIT parameter: from the catalog for cataloged data sets, from a passed data set, and from an earlier DD statement.

**Cataloged Data Sets:** When the data set is cataloged, the system obtains unit and volume information from the catalog. However, if the DD statement for a cataloged data set contains VOLUME=SER=serial-number, the system does not look in the catalog; in this case, you must code the UNIT parameter.

**Volume References to Cataloged Data Sets:** If a data set is to use the same volumes as a cataloged data set, code VOLUME=REF to refer to the cataloged data set. The system obtains unit and volume information from the catalog and places the data set on the same volumes.

**Overridden Procedure DD Statements:** When a step calls a cataloged or in-stream procedure, an overriding DD statement in the calling step statement can specify a cataloged data set in its DSNNAME parameter. If so, the overriding DD statement should nullify the UNIT and VOLUME parameters; if it does not nullify them, the system uses the UNIT and VOLUME parameters on the overridden DD statement and does not search the catalog.

## Allocation

**Passed Data Sets:** When receiving a data set passed from a previous step, omit the UNIT and VOLUME parameters. The system obtains unit and volume information from the passing step. However, if the receiving DD statement contains VOLUME=SER=serial-number, code the UNIT parameter also.

**Earlier DD Statement:** If a data set uses the volumes used for a data set in an earlier step, code a VOLUME=REF parameter to refer to the earlier DD statement. The system obtains the unit and volume information from the earlier DD statement. Therefore, you can omit the UNIT parameter. However, to make the system assign more devices or to influence device allocation, code the UNIT parameter. The system uses the coded UNIT parameter, if it requests a subset of the unit type in the referenced DD statement. Otherwise, the system ignores it.

### Unit and Volume Affinity

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. When two data sets share one volume, the data sets have **volume affinity**.

**Explicit Unit Affinity:** To reduce the number of devices for a step, request that an existing data set be assigned to the same device(s) assigned for an earlier DD statement in the same step.  
Code:

```
//ddname DD UNIT=AFF=ddname,...
```

**Implied Unit Affinity:** Implied unit affinity exists among the volumes for one data set when the DD statement requests more volumes than devices.

**Warning:** If all of the following conditions are present, the data set on a DD statement requesting unit affinity is written over by the data set on the referenced DD statement:

- The referenced DD statement makes a nonspecific volume request.
- The data set requesting unit affinity is opened before the referenced data set.
- The tape is not unloaded before the referenced data set is opened and the LABEL parameter does not request positioning of the tape to check tape labels. A tape device allocated to more than one data set is not unloaded (1) as a result of dynamic deallocation or (2) when it is closed and FREE=CLOSE is specified.

**Interaction of Unit and Volume Affinity Requests:** Unit affinity, volume affinity, and/or unit and volume affinity can exist in the same step and on the same DD statement.

If both unit and volume affinity are requested in the same step, sometimes only one affinity can be honored. Figure 15-3 on page 15-7 indicates how the system honors unit and volume affinity requests for either tape or direct access devices.

Relationship of Unit and Volume Affinity Requests	Tape	Direct Access
<p><b>All unit and volume affinity requests unrelated</b></p> <p><b>Example for Tape:</b>  //DD1 DD VOLUME=SER=A,UNIT=3420  //DD2 DD VOLUME=SER=B,UNIT=AFF=DD1  //DD3 DD VOLUME=SER=(C,D),UNIT=3420  //DD4 DD VOLUME=SER=C,UNIT=3420</p> <p><b>Example for Direct Access:</b>  //DD1 DD VOLUME=SER=A,UNIT=3340  //DD2 DD VOLUME=SER=B,UNIT=AFF=DD1  //DD3 DD VOLUME=SER=C,UNIT=3340  //DD4 DD VOLUME=SER=C,UNIT=3340</p> <ol style="list-style-type: none"> <li>Unit affinity is explicitly requested between DD1 and DD2.</li> <li>Volume affinity is implicitly requested between DD3 and DD4.</li> </ol>	<p>The system honors all unit and volume affinity requests.</p> <p>The system assigns DD2 to the same unit as DD1. The system uses volume C for DD3 and DD4.</p>	<p>The system assigns DD2 to the same unit as DD1. The system uses volume C for DD3 and DD4.</p>
<p><b>All unit and volume affinity requests related</b></p> <p><b>Example for Tape</b>  //DD1 DD VOLUME=SER=(A,D),UNIT=3420  //DD2 DD VOLUME=SER=(A,B),  // UNIT=AFF=DD1</p> <p><b>Example for Direct Access</b>  //DD1 DD VOLUME=SER=(A,D),UNIT=3340  //DD2 DD VOLUME=SER=(A,B),  // UNIT=AFF=DD1</p> <ol style="list-style-type: none"> <li>DD1 implies unit affinity because both volumes use the same unit.</li> <li>Unit affinity is explicitly requested between DD1 and DD2.</li> <li>Volume affinity is implicitly requested between DD1 and DD2, because both request volume A.</li> </ol>	<p>The system honors all unit affinity requests and ignores all volume affinity requests. Results: all volumes use the same unit.</p> <p>The system assigns DD2 to the same two units as DD1.</p> <p>Volume A resides on one 3420; volumes D and B use the other 3420.</p>	<p>The system honors all volume affinities contained in the unit affinity request; these volumes use the same unit. The other volumes in the unit affinity request use a different unit.</p> <p>The system assigns volume A for DD2 to the same 3340 as volume A for DD1. Volumes D and B use the other 3340.</p>
<p><b>Some unit and volume affinities related, some unrelated</b></p> <p><b>Example for Tape</b>  //DD1 DD VOLUME=SER=A,UNIT=3420  //DD2 DD VOLUME=SER=B,UNIT=AFF=DD1  //DD3 DD VOLUME=SER=B,UNIT=3420</p> <p><b>Example for Direct Access</b>  //DD1 DD VOLUME=SER=A,UNIT=3340  //DD2 DD VOLUME=SER=B,UNIT=AFF=DD1  //DD3 DD VOLUME=SER=B,UNIT=3340</p> <ol style="list-style-type: none"> <li>Unit affinity is explicitly requested between DD1 and DD2.</li> <li>Volume affinity is implicitly requested between DD2 and DD3.</li> </ol>	<p>The system honors all volume affinities contained in the unit affinity request; these volumes use the same unit. The other volumes in the unit affinity request use a different unit.</p> <p>The system assigns DD2 to the same unit as DD1. Volume B for DD2 and DD3 resides on a 3420. Thus, DD1, DD2, and DD3 use one 3420.</p>	<p>The system assigns volume B for DD2 and DD3 to one 3340. Volume A for DD1 uses another 3340.</p>

Figure 15-3. Unit and Volume Affinity

## Allocation

*Permanently Resident or Reserved Volumes:* If a DD statement requests a volume that is a permanently-resident or reserved volume, the system must allocate the device on which the volume is mounted, regardless of any affinities requested.

*UNIT= AFF when Requesting Extended Data Sets in a JES3 System:* In a multiple-step job in a JES3 system, if a data set is extended in an early job step to additional volumes, MVS allocates the additional devices needed. JES3 is unaware of the additional devices. If a later step requests the data set, code UNIT= AFF= ddname so that the system allocates the original and additional devices for the data set.

*Affinity for Multivolume Data Sets:* For multivolume data sets, request volume affinity if you request unit affinity. Code:

```
//ddname DD UNIT=AFF=ddname,VOLUME=REF=*.ddname,...
```

If you code only volume affinity for a multivolume data set, the following can happen:

- The system assigns the requested volumes and allocates them to a device. Thus, the device is to be shared by all the DD statements requesting volume affinity.
- The system asks the operator to mount the first volume for the referenced DD statement on the allocated device.
- At the end of the first volume, the system asks the operator to demount the first volume and mount the second volume.
- If the data set is reopened, the system asks the operator to remount the first volume on a device not used for the volume affinity request.
- When the system processes the referring DD statement, it asks the operator to mount the first volume on the device assigned to the volume affinity request. The job now enters a wait because the system has requested the first volume on two different devices.

### Specifying Device for Output Data Set

To print or punch a data set without using the job entry subsystem output service, specify the printer or punch in the UNIT parameter on the DD statement for the data set. The system allocates the device, if available, exclusively to the job; jobs cannot share output devices. Data management routines write the output from the program to the specified device.

Sending output through the job entry subsystem to a sysout data set is usually more efficient. JES uses the printers and punches for many jobs without intermixing output.

## Examples

---

```

//TEST  JOB  5675,'DEPT. 25'
//STEP1 EXEC PGM=A1
//D1    DD  DSNAME=A01DD1,DISP=(,PASS),UNIT=3330,
//      SPACE=(TRK,1),VOLUME=SER=333001
//STEP2 EXEC PGM=A2
//D2    DD  DSNAME=LIB1,DISP=OLD,UNIT=3340,
//      VOLUME=(PRIVATE,SER=123456)
//D3    DD  DSNAME=ABC,DISP=(OLD,KEEP),UNIT=AFF=D2,
//      VOLUME=SER=777777
//D4    DD  DSNAME=TAPE,DISP=OLD,UNIT=(3420-5,P,DEFER),
//      VOLUME=SER=(342001,342002,342003,342004,342005)
//D5    DD  DSNAME=DISK,DISP=(SHR,KEEP),UNIT=(,P),
//      VOLUME=SER=(333005,333008,333010)
//D6    DD  UNIT=3340,VOLUME=REF=*D2,SPACE=(TRK,(5,2))
//D7    DD  UNIT=3340,VOLUME=REF=DISK,SPACE=(TRK,(10,5))

```

- D1 defines a new data set named A01DD1. It is to be on volume 333001, which is mounted on a 3330 Disk Storage.
- D2 defines an old data set named LIB1, which resides on a private volume, 123456. The volume is mounted on a 3340 Direct Access Storage.
- D3 defines an old data set named ABC. This data set is to be kept after this step terminates. ABC is on volume 777777. This volume is to be mounted on the same 3340 device used for D2.
- D4 defines an old data set named TAPE. The data set is on the five volumes identified in the VOLUME parameter. The DEFER subparameter indicates that the five volumes are to be mounted only after the data set is opened. The P subparameter requests parallel mounting; that is, all five volumes are to be mounted at the same time on five different 3420-5 Magnetic Tape Units.
- D5 defines an old data named DISK. This data set can be shared by another job; the program only reads it. The data set is to be kept after this step. The system determines the number of devices to be allocated from the number of volume serials requested: in this case, three.
- D6 is a temporary data set, which is indicated by omission of a DSNAME parameter. The system, therefore, assumes a disposition of NEW,DELETE. The system is to place the data set on the volume used for D2 in STEP2, that is, volume 123456.
- D7 is also a temporary data set. The backward reference for volume information is to the dsname DISK, which was defined in D5 in STEP2. The system is to place this data set on the three volumes 333005, 333008, and 333010.

---

```

//STEPS EXEC PGM=TESTA
//A1    DD  UNIT=3400-5,VOLUME=SER=111111
//A2    DD  UNIT=AFF=A1,VOLUME=SER=222222

```

The system assigns one unit for both volumes. Volume 111111 is mounted first; 222222 is mounted when A2 is opened. This processing is the same for both tape and direct access.

---

## Allocation

---

```
//STEPB EXEC PGM=TESTB
//B1 DD UNIT=(3330,2),VOLUME=SER=(A,B)
//B2 DD UNIT=AFF=B1,VOLUME=SER=(C,D)
```

The system allocates two units to B1; volumes A and B are mounted. B2 gets allocated to the same two units; volumes C and D are mounted when the data set for B2 is opened.

---

```
//STEPB EXEC PGM=TESTC
//C1 DD UNIT=(3330,2),VOLUME=SER=(A,B)
//C2 DD UNIT=AFF=C1,VOLUME=SER=(C,D)
//C3 DD UNIT=3330,VOLUME=SER=B
```

STEPB shows a direct access example of volume affinity for volume B. The system allocates volumes A and C to share one unit and volumes B and D to two other units.

---

```
//STEPD EXEC PGM=TESTD
//D1 DD UNIT=(3330,2),VOLUME=SER=(E,F)
//D2 DD UNIT=AFF=D1,VOLUME=SER=(G,H)
```

STEPD is a direct access example. If volume E is currently mounted and is permanently resident or reserved, the system allocates a separate unit for volume E because it cannot be dismounted. The system allocates one unit for volume G and a second unit to be shared by volumes F and H. Therefore, three volumes are used, instead of two, because of the permanently resident or reserved attributes.

---

```
//STEPE EXEC PGM=TESTE
//E1 DD UNIT=3400-5,VOLUME=SER=(111111,222222)
//E2 DD UNIT=AFF=E1,VOLUME=SER=(222222)
```

STEPE is a tape example. The system allocates two units: one for volume 111111 and the second for volume 222222. Note that only one data set can be open on a tape volume at a time; to prevent an error when the data set for E2 is opened, the data set for E1 must be closed before E2 is opened.

---

```
//STEPF EXEC PGM=TESTF
//F1 DD UNIT=3330,VOLUME=SER=(ABCDEF,GHIJKL)
//F2 DD UNIT=AFF=F1,VOLUME=SER=(ABCDEF)
```

STEPF is a direct access example. The system ignores the volume affinity between F1 and F2. Volume ABCDEF of both DD statements uses one unit while the other volume, GHIJKL, uses a different unit.

---

```
//STEPG EXEC PGM=TESTG
//G1 DD UNIT=3400-5,VOLUME=SER=111111
//G2 DD UNIT=3400-5,VOLUME=REF=*.G1
//G3 DD UNIT=AFF=G1,VOLUME=SER=222222
```

In STEPG, G3 requests unit affinity to G1. G2 requests volume affinity to G1. Because the unit affinity request in G3 is not included in the volume affinity request, the system ignores the unit affinity request in G3. The system honors the volume affinity request. The system allocates two units: one for volume 111111 and a second for volume 222222.

---



## Device Allocation in a JES3 System

In a JES3 system, the devices and volumes for each data set are allocated by JES3 or the system.

**Device Management:** Allocation of a device depends on whether it is managed by MVS, by JES3, or jointly by JES3 and MVS. Device management is shown in the following chart.

Management	Devices
By MVS	Any devices not defined to JES3 during JES3 initialization
Jointly by JES3 and MVS	Direct access with permanently resident or reserved volumes: By JES3 for specific volume requests or for private volumes By MVS for nonspecific volume requests or for public or storage volumes
By JES3	Direct access with removable volumes Tape devices Printers Punches Graphic devices

During JES3 initialization, the installation defines how each device is to be managed. See *System Modifications* for information on MVS allocation and *SPL: JES3 Initialization and Tuning* for information on JES3 allocation.

**Device Allocation:** JES3 allocates JES3-managed devices and jointly-managed devices; JES3 performs all allocation before the job is initiated for execution. MVS allocates MVS-managed devices and jointly-managed devices; MVS performs all allocation when a step is being initiated for execution.

For a JES3-managed device, you can change the way JES3 handles allocation by coding:

```

//*MAIN  SETUP=JOB
//*MAIN  SETUP=HWS
//*MAIN  SETUP=THWS
//*MAIN  SETUP=DHWS
//*MAIN  SETUP=(stepname.ddname,...)
//*MAIN  SETUP=(stepname.procstepname.ddname,...)
//*MAIN  SETUP=/(stepname.ddname,...)
//*MAIN  SETUP=/(stepname.procstepname.ddname,...)

```

For a mass storage system (MSS) device, you can change the way JES3 handles allocation by coding:

```

//*MAIN  MSS=JOB
//*MAIN  MSS=HWS

```

**Affect of Job Class on Allocation:** The job class affects which devices can be allocated to the job. During JES3 initialization, the installation identifies the execution resources, including devices, that can be assigned to each job class.

The job class is specified by coding one of the following; if neither is coded, the system assigns the job to the installation-defined standard default class.

```

//jobname JOB acct,progname,CLASS=jobclass
//*MAIN CLASS=class-name

```

## Allocation

**Catalog Use:** For allocation, JES3 accesses the catalog at job setup time, whereas MVS accesses the catalog at step initiation time. After job setup and before step initiation, the catalog can be changed by, for example, an IBM utility, user utility, or system routine. Because JES3 and MVS access the catalog at different times, catalog changes can cause unpredictable results. Therefore, the installation should not change the catalog while jobs are being scheduled.

### Types of JES3 Setup

JES3 allocates devices in three different ways: job setup, high watermark setup, and explicit setup. The type of setup to be used is specified during JES3 initialization, but can be changed for a job by parameters on the `//*MAIN` statement.

**Job setup:** For job setup, JES3 allocates all the JES3-managed and jointly-managed devices required in the job before the job is initiated. JES3 mounts the initial volumes necessary to run all steps before the job executes. To request job setup, code:

```
//*MAIN  SETUP=JOB
//*MAIN  MSS=JOB
```

When volumes are no longer needed, they are demounted, if removable, 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 *System-Data Administration*.

**High Watermark Setup:** For high watermark setup, JES3 reserves for a job the maximum number of devices of each type needed for any one job step. JES3 premounts only some volumes before the job executes. When you must use fewer devices for a job, high watermark setup is better than job setup. To request high watermark setup, code:

- High watermark setup for tapes, direct access, graphics, printers, and punches:

```
//*MAIN  SETUP=HWS
```

- High watermark setup for tapes only, with job setup for direct access:

```
//*MAIN  SETUP=THWS
```

- High watermark setup for direct access, with job setup for tapes:

```
//*MAIN  SETUP=DHWS
```

- High watermark setup for MSS devices:

```
//*MAIN  MSS=HWS
```

When the last step that uses a device no longer needs it, JES3 deallocates it.

In the high watermark setup shown in Figure 15-4 on page 15-14, volume A is mounted for STEP1 and then demounted until needed in STEP4. Volume K is mounted for 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 directed by the user. Explicit setup requires the same number of devices as job setup. JES3 premounts volumes according to the instructions coded in:

```
//*MAIN  SETUP=(stepname.ddname,...)
//*MAIN  SETUP=(stepname.procstepname.ddname,...)
```

To request that JES3 not explicitly set up certain volumes, code:

```
//*MAIN  SETUP=/(stepname.ddname,...)
//*MAIN  SETUP=/(stepname.procstepname.ddname,...)
```

The 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. The disadvantage 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, with job setup and high watermark setup, JES3 can deallocate devices at the end of any step, if the devices are no longer needed.

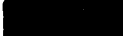


In the explicit setup shown in Figure 15-4 on page 15-14, four devices are allocated for both tape and disk instead of the three allocated using high watermark setup. The volumes to be explicitly mounted, for example, volumes A and K, are not deallocated and then remounted for the last step.

**Altering JES3 Device Allocation:** To keep JES3 from allocating devices before the first step and holding them until a later step needs them, break a multiple-step job into several smaller jobs in a dependent job net.

# Allocation

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			Direct Access			Tape			Direct Access			Tape			Direct Access									
Volumes on Devices Set Up Before 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
Job Steps <sup>2</sup>																									
STEP1 tape volume=A, B direct access volume=K, L	[Shaded]						[Shaded]						[Shaded]						[Shaded]						
STEP2 tape volume=B, C, D direct access volume=K	[Shaded]			[Shaded]			[Shaded]						C			[Shaded]			[Shaded]			[Shaded]			
STEP3 tape volume=D direct access volume=L, M, N	[Shaded]			[Shaded]			[Shaded]						M			[Shaded]			[Shaded]			[Shaded]			
STEP4 tape volume=A, E, F direct access volume=K, N, O	[Shaded]			[Shaded]			[Shaded]						A E F O K			E F			O			[Shaded]			
Total Devices Used by the Job for Setup	6 Tape						5 Direct Access						3 Tape			3 Direct Access			4 Tape			4 Direct Access			

**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* requests allocation of the minimal number of devices required to run the job.  
*THWS* requests high watermark setup for tapes and job setup for direct access.  
*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 15-4. Types of JES3 Setup**

## Allocation of Volume

Data sets on direct access and magnetic tape reside on or are written on volumes. The volumes may be permanently mounted on the device or may need to be mounted by the operator. To tell the system the volume on which an existing data set resides, make a specific volume request. To tell the system the volume on which to write a new data set, make a specific or nonspecific volume request.

**Volume Attributes:** The system assigns volumes two attributes:

- **Use attributes**, which control how volumes are allocated, are:
  - **Private:** The volume can be allocated only when its volume serial number is explicitly or implicitly specified.
  - **Public:** The volume is eligible for allocation to temporary data sets defined with a nonspecific volume request and without a PRIVATE subparameter in the VOLUME parameter.
  - **Storage:** The volume is eligible for allocation to both temporary and permanent data sets defined with a nonspecific volume request and without a PRIVATE subparameter in the VOLUME parameter. Storage volumes usually contain permanent data sets, but can be used for temporary data sets.
- **Mount attributes**, which control how or whether volumes can be demounted after being deallocated, are:
  - **Permanently resident:** The volume, which can only be direct access, cannot be demounted. Volumes that are always permanently resident are all volumes that cannot be physically demounted, the IPL volume, and the volume containing system data sets. Permanently resident volumes have any use attribute.
  - **Reserved:** The volume remains mounted until the operator issues an UNLOAD command. Volumes that should be reserved are volumes that are used frequently by many jobs. Reserved, direct access volumes can have any use attribute; reserved, tape volumes can be only private or public.
  - **Removable:** The volume is neither permanently resident nor reserved. Removable volumes can be demounted after their last use in a job. Removable volumes can be only private or public.

For more information on attributes, see *System Modifications*.

### Specific Volume Requests

Make a specific volume request by coding:

```
//ddname DD VOLUME=SER=serial-number
//ddname DD VOLUME=REF=dsname
//ddname DD VOLUME=REF=*.ddname

//ddname DD DSNAME=passed data set
//ddname DD DSNAME=cataloged data set
```

## Allocation

For passed or cataloged data sets, the system obtains the volume serial numbers from the passed data set information or from the catalog. In these cases, do not code a SER or REF subparameter in a VOLUME parameter; other VOLUME subparameters can be coded.

**How the System Satisfies Specific Volume Requests:** In the following cases, the system satisfies a request for a specific volume with a volume that is already mounted:

- The requested volume is permanently resident or reserved. The system assigns the volume regardless of whether public or private use was requested; the volume retains its original use attribute of public or private.
- The requested volume is a removable direct access volume that can be shared and is being used by a concurrently executing step. If the request would make the volume unable to be shared, the system assigns the volume only after all other steps using it terminate.
- The requested volume is a removable direct access volume that is mounted but not allocated. The volume is assigned a use attribute of private if the VOLUME parameter specifies PRIVATE; otherwise, the volume is for public use.
- The requested volume is a scratch tape volume that is mounted but not allocated. The tape is assigned a private attribute if the request is for a permanent data set or if the VOLUME parameter specifies PRIVATE; otherwise, the volume is for public use.

### Nonspecific Volume Requests

Make a nonspecific volume request for a new data set that can be assigned to any volume or volumes. To make a nonspecific volume request, either:

- Omit the VOLUME parameter.
- Code a VOLUME parameter but omit a SER or REF subparameter.

**How the System Satisfies Nonspecific Volume Requests:** The system satisfies a request for a nonspecific volume as follows:

#### **Request for private volume for temporary or permanent data set**

For direct access or tape, the system always asks the operator to mount a volume. The operator should mount a volume containing only unused space so that the owner can control all the space on the volume. Once mounted, the volume is assigned the attribute of private.

#### **Request for public volume for temporary data set**

For direct access, the system assigns a public or storage volume that is already mounted or, if no space is available, the system asks the operator to mount a removable volume. If the system selects a mounted, public volume, it remains public. If the operator mounts a volume, it is designated a public volume.

For tape, the system assigns any available, mounted, tape volume; if none is available, the system asks the operator to mount a tape volume. Once mounted, the volume is assigned the use attribute of public.

Assigning an available, mounted volume could result in the loss of user data. However, if the tape volumes are labeled and the LABEL parameter specifies the label type, loss of data is usually prevented because the system checks the first record of the tape when opening the data set.

## Request for public volume for permanent data set

For direct access, 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.

For tape volume, the system treats the request as a nonspecific volume request for a private volume.

## Private Volumes

The system assigns a removable volume a use attribute of private if **any** one of the following is true:

- The VOLUME parameter contains the PRIVATE subparameter.
- The DD statement requests a specific volume.
- The DD statement requests a permanent data set; that is, the data set does not have a system-generated data set name and the DISP parameter does not specify DELETE.

To make a direct access volume private, code:

```
//ddname DD VOLUME=PRIVATE
//ddname DD VOLUME=SER=xxxxxx
//ddname DD VOLUME=REF=*.ddname
//ddname DD DSNAME=permanentds,DISP=(,KEEP)
//ddname DD DSNAME=permanentds,DISP=(,CATLG)
```

To make a tape volume private, specify or obtain the the volume serial number; because the request is for a specific volume, the system automatically makes the tape volume private.

*Using Private Volumes:* To use a private volume, you must give the system the serial number: the DD statement must specify the serial number or obtain it from a previous DD statement through a VOLUME=REF parameter.

The system cannot assign a nonspecific volume request to a private volume. Therefore, if you request a private volume, you will be the only one using that volume, unless another job makes a specific volume request for that volume.

## Public Volumes

The system assigns a removable volume a use attribute of public when **all** of the following are true:

- The VOLUME parameter does **not** contain a PRIVATE subparameter.
- The DD statement does **not** request a specific volume.
- The DD statement requests a temporary data set; that is, no name is specified for the data set name or the disposition is DISP=(NEW,DELETE) or a DISP parameter is omitted to imply a new data set to be deleted.

# Allocation

## Volume Affinity

To use fewer volumes, assign data sets to the same volume. Data sets on the same volume have **volume affinity**.

Volume affinity influences the allocation of devices. A request for volume affinity with another data set can make the system modify a request for a specific number of units in the unit count subparameter of the UNIT parameter.

**Explicit Volume Affinity:** To request that a new data set be assigned to the same volume(s) as another data set, code:

```
//ddname DD VOLUME=REF=dsname  
//ddname DD VOLUME=REF=*.ddname  
//ddname DD VOLUME=REF=*.stepname.ddname  
//ddname DD VOLUME=REF=*.stepname.procstepname.ddname  
//ddname DD VOLUME=REF=*.procstepname.ddname
```

Use the first form to reference a cataloged or passed data set. Use the other forms to reference a DD statement earlier in the job.

**Implicit Volume Affinity:** To request volume affinity implicitly, specify the serial number(s) of the volume(s) containing another data set.

## Multivolume Data Sets

**Number of Volumes:** When creating or extending a data set, request the maximum number of volumes that may be required. Indicate the number in the volume-count specified in the VOLUME parameter or by the number of serial numbers implicitly or explicitly specified.

If you make a specific volume request for more volumes than units, the system automatically indicates that the volumes allocated to the same unit cannot be shared.

If you request multiple direct access volumes in a JES3 system, they must be either all mountable or all permanently mounted; a mixture is not allowed.

**Parallel Mounting:** For some jobs, all requested volumes must be mounted before the data set can be used. For these jobs, request as many units as volumes or request parallel mounting by coding P in the UNIT parameter.

**Processing Order:** When reading or adding to an existing multivolume data set, you can tell the system to begin processing with other than the first volume by coding:

```
//ddname DD VOLUME=(,,,volume-sequence-number),...
```

## Volumes Required per DD Statement

The maximum number of tape volumes or direct access volumes required to satisfy any DD statement is the greater of:

- volume-count specified in the VOLUME parameter:

```
//ddname DD VOLUME=(,,,volume-count),...
```

- number of serial numbers implicitly or explicitly specified



The number of serial numbers implicitly or explicitly specified is:

- The number of volume serials in the `VOLUME=SER` subparameter:

```
//ddname DD VOLUME=SER=(serial-number,serial-number,...),...
```

- The number of volume serials obtained through `VOLUME=REF`, if coded:

```
//ddname DD VOLUME=REF=dsname  
//ddname DD VOLUME=REF=*.ddname
```

- The number of volume serials obtained from passed data set information, if the `DD` statement is receiving a passed data set from a prior step. The receiving `DD` statement must not specify `VOLUME=SER` or `VOLUME=REF`; if it does, the system obtains the number from the `VOLUME` parameter.
- The number of volume serials obtained from the catalog, if the `DD` statement requests an existing, cataloged data set. The `DD` statement must not specify `VOLUME=SER` or `VOLUME=REF`; if it does, the system obtains the number from the `VOLUME` parameter. Also, the data set must not be passed from a prior step.
- The number of volume serials minus the volume sequence number plus one, if the `DD` statement requests an existing data set and specifies a volume sequence number. For example, if the `DD` statement specifies eight volume serial numbers and a volume sequence number of four, the system uses five volume serials:  $8 - 4 + 1 = 5$ . The first three volume serials are not used; the first volume that the system allocates is the fourth volume.
- The number of volume serials implied by the unit count in the `UNIT` parameter, if (1) the unit count is higher than the calculated number of volume serials or (2) the `DD` statement makes a nonspecific volume request for a new data set on direct access for public use.

When the volume count or unit count require more volumes than the number specified in `VOLUME=SER` or obtained from `VOLUME=REF`, passed data set information, or the catalog, the system assumes that the requests are for nonspecific volumes.

### Mass Storage Volume Groups

The 3850 Mass Storage System can contain up to 4,720 mass storage volumes. Each volume is requested by coding `UNIT=3330V`. Mass storage volumes reside on virtual direct access devices. The system allocates mass storage volumes in the same way as direct access devices and volumes.

Using the mass storage system service, the installation assigns mass storage volumes to groups; each group is an installation-defined subset of all mass storage volumes. The installation can define as many groups as necessary. One group is standard in all systems: `SYSGROUP`. The installation assigns each mass storage volume to a user group, to `SYSGROUP`, or to no group.

See the *Mass Storage System (MSS) Services General Information* for more information.

**Nonspecific Volume Requests for Mass Storage Volumes:** When defining a new data set for a mass storage volume with a nonspecific volume request, specify a group by coding:

```
//ddname DD UNIT=3330V,MSVGP=id,...
```

## Allocation

From the identified group, the system selects a volume with enough space to satisfy the space requirements of the DD statement.

When you code the MSVGP parameter, use the VOLUME parameter to specify a volume count, if more than one volume is needed. Do not code VOLUME=PRIVATE; MSVGP implies private.

If a DD statement makes a nonspecific request for an MSS volume but does not contain an MSVGP parameter, the system does the following:

- For a data set needing a private volume, assigns the data set to the default group, SYSGROUP.
- For a temporary data set not needing a private volume, assigns a public or storage MSS volume that is already mounted, if available. If not available, treats the request like a nonspecific volume request for a private volume.
- For a permanent data set not needing a private volume, assigns a public MSS volume, if available. If not available, treats the request like a nonspecific volume request for a private volume.

**Specific Volume Requests for Mass Storage Volumes:** Specific volume requests for MSS volumes are treated the same as specific volume requests for direct access volumes.

**Cataloging Data Sets on MSS:** Because MSS volumes cannot be mounted and demounted, the installation should catalog all permanent data sets on MSS volumes. Cataloging helps maintain data integrity. DD statements should always request these data sets by their cataloged name coded in the DSNNAME parameter. Do not code a VOLUME parameter.

The catalog must be referenced when extending an existing data set to additional volumes. Using the cataloged name in the DSNNAME parameter lets the system determine all volumes on which the data set currently resides before it selects the new volume. To make sure that the data set is extended correctly, specify parallel mounting by coding:

```
//ddname DD UNIT=(3330V,P)
```

**Placing Data Sets on Different MSS Volumes:** Two data sets may need to be placed on different MSS volumes. For example, a program loop reads a record from a master data set, processes it, then writes a record to a new data set; the program executes faster if the two data sets are on different volumes. Request that data sets be allocated to different volumes by coding:

```
//ddname DD UNIT=3330V,MSVGP=(id,ddname),...
```

### Examples

---

For examples of volume allocation, see “Examples” on page 15-9.

---

## Allocation of Direct Access Space

You must request space for every non-VSAM data set being created on a direct access volume. To tell the system how much space is needed and let the system assign the tracks, code:

```
//ddname DD SPACE=(TRK,(primary-qty,second-qty,directory or index)),...
//ddname DD SPACE=(CYL,(primary-qty,second-qty,directory or index)),...
//ddname DD SPACE=(blklgth,(primary-qty,second-qty,directory or index)),...
```

To tell the system the specific tracks to assign to the data set, code:

```
//ddname DD SPACE=(ABSTR,(primary-qty,address,directory or index)),...
```

## Requesting System Assigned Space

Letting the system assign the specific tracks is easiest and most frequently used. Specify only how the space is to be measured — in tracks, cylinders, or blocks — and how many of those tracks, cylinders, or blocks are required.

**Requests for Blocks:** It is easiest to specify an average block length: the system allocates 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 you request space in terms of average block length, the system allocates tracks to contain the request. However, if you code ROUND as the last subparameter in the SPACE parameter, the system allocates the smallest number of cylinders needed to contain the request.

**Requests for Tracks or Cylinders:** 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 Administration Guide*.

Cylinder allocation allows faster input/output of sequential data sets than does track allocation.

## How the System Satisfies the Primary Space Request

**Space on One Volume:** Enough space must be available on one volume to satisfy the primary request. If not, the system terminates the job or searches another volume, depending on the type of volume request made:

**Specific volume request:** If the first volume specified does not have enough space available, the job is terminated. When extending a multivolume data set, if enough space is not available to satisfy the secondary allocation on the second volume specified, the job is terminated.

**Nonspecific volume request:** If the first volume chosen by the system does not have enough space available, the system chooses another volume and continues to search for space,

## Allocation

asking for volumes to be mounted if necessary. The system continues to search for space until it finds a volume with enough space or the operator cancels the job.

*Note:* For a new indexed sequential data set, if the first volume chosen by the system does not contain enough space for the request, the system does not try to find space on another volume, if the request is as follows:

- A request for multiple volumes or units.
- A request is for the second, third, or subsequent DD statement used to define the data set.

*Extents:* The system tries 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.

*System Assigned Space Requests with User Labels:* If user labels are specified, LABEL=(,SUL), the system allocates up to four noncontiguous extents of space. The system allocates, separately from the primary quantity, one track for user labels. This one track is considered an extent.

### How the System Satisfies the Secondary Space Request

For many data sets, the primary quantity does not need to be big enough for the entire data set. Code a **secondary quantity** to be used only if the data set exceeds its originally allocated space.

*Note:* BDAM data sets cannot be extended.

*Volume for Secondary Space for NEW or MOD Data Set:* For data sets whose disposition is NEW or MOD, the system allocates this space 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, less the number of extents for the primary quantity and user label space, have been allocated to the data set.

Then, the system allocates the secondary quantity on another volume only if the DD statement requested more than one volume in the VOLUME parameter or, for a specific volume request, requested more volumes than devices.

If the DD statement makes a nonspecific volume request and the system could possibly allocate a permanently resident volume, code PRIVATE in the VOLUME parameter.

*Volume for Secondary Space for OLD Data Set:* When allocating a secondary quantity for a data set whose status is OLD, that is, an existing data set being written over or a preallocated data set, the system checks for a next volume. If a next volume exists, the system looks for a secondary quantity already allocated in it. If the system finds a secondary quantity, the system uses that space. If the system finds no space already allocated, the system allocates the secondary quantity on that next volume. If a next volume does not exist, the system allocates the secondary space on the current volume.

**Secondary Request Only for Current Execution:** 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.

**Secondary Requests in Blocks:** If you request space in terms of average block length, supply 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 in the data control block to compute how many additional tracks to allocate.

## Directory Space for Partitioned Data Sets

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 the partitioned data set. It consists of a 256-byte record for each member. The third quantity in the SPACE parameter must specify how many records the directory is to contain.

The directory is written at the beginning of the primary space. Request enough directory space to allow for growth of the data set. You cannot lengthen the directory once the data set is created. If the directory runs out of space, you must recreate the data set.

For a complete description of the directory, including details on member entries to enable you to compute how many records to request, see *Data Administration Guide*.

## System Assigned Space Requests for Indexed Sequential Data Sets

For an indexed sequential data set, space must be requested in cylinders.

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, request index space as the third quantity in the SPACE parameter. The system determines if the third quantity is for a directory or an index from the DCB parameter on the DD statement: DCB=DSORG=IS or DCB=DSORG=ISU must be specified when defining an indexed sequential data set. The system adds the index quantity to the primary quantity when allocating space.

## Example

---

```
//ALLO      JOB   (3416,354),STONER,MSGLEVEL=1,MSGCLASS=C
//STEP1     EXEC  PGM=TESTSYS0
//DD1       DD   UNIT=3350,DISP=(,PASS),SPACE=(TRK,(10,5))
//DD2       DD   UNIT=3330,DISP=(,PASS),SPACE=(TRK,(10,5))
//SYSABEND  DD   SYSOUT=L
//STEP2     EXEC  PGM=TESTSYS0
//DD3       DD   DSNAME=*.STEP1.DD1,DISP=(OLD,DELETE,DELETE)
//DD4       DD   VOLUME=REF=*.STEP1.DD2,SPACE=(TRK,(3,1)),UNIT=3330
//SYSABEND  DD   SYSOUT=L
```

The first step requests space for two temporary data sets. The second step refers to these data sets for volume information. The space requested for DD1 and DD2 in STEP1 is 10 primary and 5 secondary tracks and for DD4 in STEP2 3 primary and 1 secondary tracks.

---

## Allocation

### Requesting Specific Tracks

Requesting that the system allocate specified tracks to a data set is the most stringent request for space. If any of the requested tracks on the volume are occupied, the space cannot be allocated and the job is terminated.

Certain uses of certain devices can require that specific tracks be requested. For example, specific tracks must be allocated to position a data set under the fixed heads of a 3348 Model 70F Data Module (cylinders 1-5).

**Specific Track Requests with User Labels:** If user labels are specified, LABEL=(,SUL), the user labels are placed on a user label track. This track is the first in the space requested.

### Specific Track Requests for Indexed Sequential Data Sets

If defining an indexed sequential data set, 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

---

```
//DDEX DD SPACE=(ABSTR,(1,1)),...
```

This example allocates one track for a data set: specifically, the second track on a volume.

---

## Allocation of Virtual I/O

Temporary data sets can be handled by a facility called virtual input/output (VIO). VIO data sets reside in the paging space; but, to the problem program and the access method, the data sets appear to reside on a direct access storage device.

VIO provides two advantages:

- VIO speeds reading or writing of a data set. All reading and writing operations are done at the speed of main storage access rather than at the speed of I/O to a device.
- The virtual data set does not occupy space in the user's private area. Thus, unlike a large data area in a program, a virtual data set does not use up program space.

VIO cannot be used for permanent data sets, indexed sequential data sets, VSAM data sets, or empty input data sets.

**Requesting VIO:** To request a VIO data set, code a DD statement as follows:

- The DSNNAME parameter can be coded or omitted. If coded, it must specify a temporary data set:

```
DSNNAME=&&dsname  
DSNNAME=&&dsname(member)
```

- The DISP parameter can be coded or omitted. If coded, it must specify:

```
DISP=(NEW,DELETE)
DISP=(NEW,PASS)
DISP=(,PASS)
```

- Code a UNIT parameter. It must specify a VIO unit name. During system generation, the installation must define new and/or existing unit names as VIO; the installation should maintain a list of the VIO unit names.

The unit count subparameter is ignored, if coded.

- The VOLUME parameter can be coded or omitted. If coded, do not specify volume serial numbers.
- The SPACE parameter can be coded or omitted. If coded, the parameter can request up to the size of the simulated volume. The system will allocate as the primary quantity plus 15 secondary quantities an entire simulated volume.

If the requested primary quantity 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, the system allocates up to one volume. When allocating by average block length for a VIO data set, the secondary request is computed using the average block length specified in the SPACE parameter.

If the SPACE parameter is omitted, the system uses a default value: 10 primary and 50 secondary blocks, with an average block length of 1000.

- The DCB parameter can be coded or omitted. If coded, do not specify IS or ISU in the DSORG subparameter.

The system will allocate a VIO data set request to actual direct access storage if the DD statement contains unacceptable parameters; however, if the primary quantity is too big, the system terminates the job.

## Examples

---

```
//EX1 DD UNIT=VIO
//EX2 DD DSNAME=&&TEMPDS,UNIT=SYSDA
//EX3 DD DSNAME=&&TEMPDS(MEM1),UNIT=VIRT3
//EX4 DD DSNAME=&&MYDS,UNIT=VIO,SPACE=(360,(5,30)),
//      DISP=(,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=360)
```

In these examples, the system assigned during system generation the group names VIO, SYSDA, and VIRT3 as eligible for VIO processing.

---

## Backward References to VIO Data Sets

If a DD statement defines a temporary data set and refers in a VOLUME=REF parameter to a DD statement for a VIO data set, the system assigns the data set to external page storage as a VIO data set.

If a DD statement requests unit affinity to a VIO data set but does not define a temporary data set, the system allocates the data set to the VIO unit but does not assign it VIO status.

## Allocation

**Examples:** The examples assume that the installation defined during system generation the group name SYSDA and the device type name 3330 as eligible for VIO processing. Except where noted, all of the following DD statements cause allocation of VIO data sets.

---

```
//DD1 DD UNIT=SYSDA
```

---

```
//DD2 DD UNIT=3330
```

---

```
//DD3 DD DSNAME=&&A,DISP=(NEW),SPACE=(CYL,(30,10)),UNIT=SYSDA
```

---

```
//DD1 DD UNIT=SYSDA  
//DD2 DD VOLUME=REF=* .DD1
```

---

```
//DDA DD UNIT=SYSDA  
//ddb DD VOLUME=REF=* .DDA,UNIT=3330
```

---

```
//DD1 DD UNIT=SYSDA  
//DD2 DD DSNAME=NONTEMP,DISP=(,KEEP),  
// VOLUME=REF=* .DD1,SPACE=(CYL,10)
```

In this example, the data set defined in DD1 is assigned to external page storage for VIO processing. Because DD2 defines a permanent data set, the system assigns it to direct access storage.

---

```
//DD1 DD UNIT=SYSDA  
//DD2 DD DSNAME=&&TEMP,VOLUME=SER=665431,  
// SPACE=(CYL,10),UNIT=AFF=DD1
```

In this example, the data set defined in DD1 is assigned to external page storage for VIO processing. Because DD2 specifies a volume serial number, the system assigns it to direct access storage.

---



---

```

//REGJOB      JOB  3344,'DEPT. 28'
//ASM         EXEC PGM=IFOX00
.
.
//ASM.SYSGO   DD   DSNAME=&&OBJ,UNIT=VIO,DISP=(NEW,PASS)
//LKED        EXEC PGM=IEWL
//SYSLIN      DD   DSNAME=&&OBJ,DISP=(OLD,DELETE)
//           DD   DDNAME=SYSIN
//SYSLMOD     DD   DSNAME=&&LOAD(A),DISP=(NEW,PASS),UNIT=VIO,
//           DCB=DSORG=PO,SPACE=(TRK,(5,5,1))
.
.
//GO          EXEC PGM=*.LKED.SYSLMOD

```

VIO data sets are passed in the same way as conventional data sets. This example shows the DD statements for VIO data sets in a job whose steps compile and link edit a program and then execute that program. The three VIO data sets are defined in the statements ASM.SYSGO, SYSLIN, and SYSLMOD.

*Note:* The SPACE parameter must appear on the //SYSLMOD DD statement to make sure that directory space is allocated.

---

## Allocation with Deferred Volume Mounting

A step can include a data set that the program might not use. To ask the system not to mount the volume for the data set until the data set is opened, code:

```
//ddname DD UNIT=(xxxx,,DEFER),...
```

Deferred mounting can save the operator time.

### Example

---

```
//MYDS DD DSNAME=DATA5,UNIT=(TAPE,,DEFER)
```

---

## Allocation with Volume Premounting in a JES2 System

In a JES2 system, to identify volumes that the operator must mount before the job is executed, code:

```
/*SETUP serial-number,...
```

When the job enters the system, JES2 issues a message to the operator console to ask the operator to mount the identified volumes. JES2 places the job on hold until the operator mounts the volumes, then releases the job.

## Allocation

### Example

---

```
/*SETUP 223344,556677,889900
```

---

## Dynamic Allocation

Dynamic allocation allows a job to acquire resources as they are needed and release them immediately after use. The resources are a ddname-data set combination with its volumes and devices.

One reason to use dynamic allocation is that you may not know all of the device requirements for a job before 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 release them immediately after use.

To tell the system the number of resources to be held in anticipation of reuse, code:

```
//stepname EXEC PGM=x,DYNAMNBR=n
```

The system uses the sum of this number and the number of DD statements in the step to establish a control limit for tracking resources that it is holding in anticipation of reuse.

For more information on dynamic allocation, see *SPL: System Modifications*.

### Example

---

```
//PROS JOB 1585,SALLYJ,CLASS=A,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
.
/*
```

- The JOB statement specifies that this job will be processed in class A and in performance group 70.
- The control limit is the sum of the number of DD statements coded and the value coded in the DYNAMNBR parameter;

3 DD statements + 4 = 7

If this control limit is reached and another dynamic allocation is requested, the request is not honored unless resources can be deallocated so that the control limit is not exceeded.

- When OUT1 is closed, it is immediately ready for printing.
-

## Chapter 16. Processing Control

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Processing control</b>					
by suppressing processing	DUMMY NULLFILE on DSNAME				
by postponing specification	DDNAME				
with checkpointing	CHKPT SYSCKEOV DD				
by subsystem	SUBSYS CNTL		CNTL ENDCNTL		
by TCAM job or task	QNAME				

Figure 16-1. Processing Control Task for Requesting Data Set Resources

### Processing Control by Suppressing Processing

To suppress processing of a data set, assign it a dummy status by coding either of the following:

```
//ddname DD DUMMY,...
//ddname DD DSNAME=NULLFILE,...
```

The system ignores all parameters other than DUMMY or DSNAME=NULLFILE and DCB. 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, code this information in the DD DCB parameter.

**Effect of Dummy Data Set:** For a dummy data set, the system bypasses all input/output operations, does not allocate devices or storage to the data set, and does not perform disposition processing.

## Processing Control

**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 writes to the dummy data set, the request is recognized but no data is transmitted. VSAM supports dummy data sets for both read and write processing. BSAM and QSAM support requests to write to a dummy data set. If any other access method is used, the job is terminated.

**Use of Dummy Data Sets:** When testing a program, you can suppress writing of an output data set by defining it as a dummy data set. This would forestall printing a data set until you are sure it contains meaningful output.

To save processing time, you might not want a data set to be processed every time the job is executed. For example, you might want to skip reading a data set that is used only once a week.

**Nullifying a Dummy Data Set:** When the data set is to be processed, replace the DD statement that specified the dummy data set with a DD statement containing the parameters required to define the data set. When a procedure DD statement specifies a dummy data set, nullify it by coding the DSNNAME parameter on the overriding DD statement and assigning a data set name other than NULLFILE.

### Examples

---

```
//EXA DD DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//      UNIT=3211
//EXB DD DSNNAME=NULLFILE,UNIT=DISK,VOLUME=SER=165789,
//      DISP=OLD
//EXC DD DUMMY,DISP=OLD
```

---

## Processing Control by Postponing Specification

To postpone specification of a data set, reference a later DD statement by coding:

```
//ddname DD DDNAME=ddname
```

**How the System Postpones Data Set Definition:** When the system encounters a DD statement with a DDNAME parameter, it saves the ddname and, temporarily, the name in the DDNAME parameter; the system uses the DDNAME name to relate the statement to a later DD statement. When the system finds a statement whose ddname has been temporarily saved, it does the following:

- It uses the parameters on the statement with the matching ddname to define the data set.
- It associates these parameters with the name of the statement that contained the DDNAME parameter.
- It stops saving the name from the DDNAME parameter.

**References to the Data Set:** The system associates the ddname of the statement that contains the DDNAME parameter with the data set definition. The system does not use the ddname of the later statement that actually 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.

**Concatenating DD Statements when DDNAME is Specified:** To concatenate data sets to a data set defined with a DDNAME parameter, the unnamed DD statements must follow the DD statement that contains the DDNAME parameter, not the DD statement that defines the data set.

**Use of Postponing Specification:** Use the DDNAME parameter in cataloged procedures to postpone defining an in-stream data set until a job step calls the procedure. Procedures cannot contain DD statements that define in-stream data sets and cannot contain in-stream data.

Use the DDNAME parameter in a job step that calls a procedure to postpone defining in-stream data until the last overriding DD statement for a procedure step. Overriding DD statements must appear in the same order as the DD statements in the procedure and any in-stream data sets must appear last in a calling step.

### Examples

---

```
//XYZ DD DDNAME=PHOB
      .
      .
//PHOB DD DSN=NIN,DISP=(NEW,KEEP),UNIT=3400-5
```

From DD statement XYZ, the system saves XYZ and, temporarily, PHOB. Until the system encounters the ddname PHOB, it treats the data set for XYZ as a dummy data set.

When the system reads DD statement PHOB, it uses the DSN, DISP, and UNIT values to define the data set named NIN. The system also associates this information with DD statement XYZ. The system stops saving ddname PHOB. The data set is now defined as if you had coded:

```
//XYZ DD DSN=NIN,DISP=(NEW,KEEP),UNIT=3400-5
```

---

```
//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.

---

## Processing Control

---

```
//DDA DD DDNAME=DEF
// DD DSN=A.B.C,DISP=OLD
// DD DSN=SEVC,DISP=OLD,UNIT=3350,VOL=SER=52226
.
.
//DEF DD *
data
/*
```

This example shows correct concatenation when a DDNAME parameter is coded.

---

## Processing Control with Checkpointing

To write a checkpoint when the system reaches an end of volume while processing a multivolume input or output data set, code:

```
//ddname DD CHKPT=EOV,...
```

The system writes checkpoints for all volumes but the last. The data set must be a multivolume QSAM or BSAM data set. Checkpoints are not written for single-volume QSAM or BSAM data sets or for ISAM, BDAM, BPAM, or VSAM data sets.

The system writes the checkpoints in a SYSCKEOV data set. A SYSCKEOV DD statement must be specified in a step with a DD statement that contains CHKPT and again when the step is restarted from a checkpoint written in the data set.

### Examples

---

```
//S1 EXEC PGM=A, RD=R
//D1 DD DSN=OUT1, UNIT=(DISK, 3), DISP=(NEW, CATLG),
// SPACE=(400, (50, 10), VOLUME=(PRIVATE, , , 3), CHKPT=EOV
//SYSCKEOV DD DSN=CK1, UNIT=3350, DISP=(MOD, KEEP),
// SPACE=(CYL, 30, , CONTIG)
```

---

## Processing Control by Subsystem

### Requesting Subsystem

To ask a subsystem to process a data set and to specify parameters for the subsystem, code:

```
//ddname DD SUBSYS=subsystem-name,...
//ddname DD SUBSYS=(subsystem-name,subparameter,...),...
```

The subsystem processes the subparameters according to its own rules.

When you specify the SUBSYS parameter, the subsystem may alter the significance of certain DD statement parameters. For details, see the documentation for the subsystem.

If you specify the DUMMY parameter, MVS invokes the subsystem to check the syntax of subsystem subparameters. 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.

#### *Example*

---

```
//EXSUB DD DSNAME=MYSET,DISP=OLD,SUBSYS=(PRO3,34,92)
```

---

### Program Control Statements for a Subsystem

To specify control information for a subsystem, code:

```
//stepname EXEC PGM=x
//label CNTL *
.
(program control statements)
.
// ENDCNTL
//ddname DD SUBSYS=subsystem-name,CNTL=*.label
```

Program control statements supply control information for the subsystem.

#### *Example*

---

```
//S1 EXEC PGM=REPT
//ABC CNTL *
//PGC PRINTDEV BUFNO=2-,PIMSG=YES
// ENDCNTL
//DD1 DD SUBSYS=PSF,CNTL=*.ABC
```

---

## Processing Control

### Processing Control by TCAM Job or Task

To define a data set of telecommunications access method (TCAM) messages and to ask a TCAM job or started task to process the data set, code:

```
//ddname DD QNAME=procname,...  
//ddname DD QNAME=procname.tcamname,...
```

For more information, see *Advanced Communications Function for TCAM, Version 2, Installation Reference*.

#### Examples

---

```
//DSA DD QNAME=MES34.TJOB,DCB=(RECFM=FB,LRECL=80,BLKSIZE=320)  
//DSB DD QNAME=MES78.TJOB
```

---



## Chapter 17. End Processing

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
End processing					
deallocation	FREE				
disposition of data set	DISP  RET and EXPDT on LABEL				
release of unused direct access space	RLSE on SPACE				
disposition of volume	RETAIN and PRIVATE on VOLUME				

Figure 17-1. End Processing Task for Requesting Data Set Resources

### Deallocation End Processing

The system deallocates data sets and their associated volume and devices at the end of a job step or at the end of the job.

**Dynamic Deallocation:** To deallocate a data set while a step is still executing, code:

```
//ddname DD FREE=CLOSE,...
```

Use FREE=CLOSE to allow the system to reallocate a volume or device that is used frequently in the system.

## End Processing

### Example

---

```
//DD1 DD DSNAME=DS6,DISP=OLD,UNIT=TAPE,VOLUME=SER=111111,FREE=CLOSE
```

---

## Disposition End Processing of Data Set

### Disposition Controlled by DISP Parameter

The system processes a data set after its use depending on how the step terminates:

- **Normal termination disposition:** To delete, keep, pass, catalog, or uncatalog the data set when the step terminates normally, code:

```
//ddname DD DISP=(,DELETE),...  
//ddname DD DISP=(,KEEP),...  
//ddname DD DISP=(,CATLG),...  
//ddname DD DISP=(,UNCATLG),...  
//ddname DD DISP=(,PASS),...
```

- **Abnormal termination or conditional disposition:** To delete, keep, catalog, or uncatalog the data set if the step terminates abnormally, code:

```
//ddname DD DISP=(,,DELETE),...  
//ddname DD DISP=(,,KEEP),...  
//ddname DD DISP=(,,CATLG),...  
//ddname DD DISP=(,,UNCATLG),...
```

You should consider coding an abnormal termination disposition every time you create or use a data set. This disposition can be used to keep data sets after a program fails, when they might be needed to determine the cause of the failure. This 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. Then the failing job can be rerun without an intervening clean-up job.

**Effect of Abnormal Termination During Execution:** When a step abnormally terminates but is not automatically restarted, its data sets are disposed of as specified by the abnormal termination disposition. If an abnormal termination disposition is not specified, the normal termination disposition is processed.

**Effect of Abnormal Termination During Allocation:** If a job step fails during step allocation, the system disposes of the data sets as follows:

- Deletes a data set being created in the step.
- Keeps a data set that existed before the step.

**Effect When No Abnormal Termination Disposition is Coded:** If a DD statement in an abnormally terminating step requests a data set that was cataloged or kept in an earlier step and if the statement does not specify an abnormal termination disposition, the system uses the disposition specified in the earlier step.

**Effect of Device Type on Disposition:** The system handles disposition differently for data sets on direct access and on tape. A direct access volume contains a volume table of contents (VTOC). A VTOC consists of control blocks describing the non-VSAM data sets and available space on the volume.

### Deleting a Data Set

Specifying DELETE requests that the data set's space on the volume be released at termination of the step:

- If the data set is on a public tape volume, the tape is rewound. The volume is available for use by other job steps.
- If the data set is on a private tape volume, the tape is rewound and unloaded. The system issues a KEEP message.
- If the data set is on a private direct access volume, the control block describing the data set is removed from the VTOC. The space on the volume is available to other data sets.

**Unexpired Expiration Date:** In one case, however, a data set on a direct access volume is not deleted: If a data set previously existed and has an unexpired expiration date, an abnormal termination disposition of DELETE does not delete the data set if the step abnormally terminates.

**Cataloged Data Sets:** 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 serial number was not coded or referenced on the DD statement. If the system did not obtain volume information from the catalog, the data set is deleted but its entry remains in the catalog.

If an error occurs while the system is deleting a cataloged data set, its entry remains in the catalog. The data set itself is or is not deleted, depending on when the error occurs.

To delete an entry from a VSAM catalog, use the DELETE command as described in *VSAM Administration Guide*. Using the DELETE command makes the space occupied by the data set available for reallocation. To delete catalog entries for data sets that are not cataloged in a VSAM catalog, use the UNCATLG statement of IEHPROGM as described in *Data Administration: Utilities*.

**Temporary Data Sets:** DELETE is the only valid abnormal termination disposition for a temporary data set. 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 later step or job requests that the data set be deleted or cataloged or until after an expiration date or retention period, if specified.

For data sets on direct access, the entry in the VTOC describing the data set and the data set itself are kept. For data sets on tape, the volume is rewound and unloaded, and a KEEP message is issued to the operator.

# End Processing

## Cataloging a Data Set

Catalog a non-VSAM data set by specifying CATLG as the disposition. The system keeps the data set and creates an entry pointing to it in one of the following:

- The system master catalog, if the step or job does not specify a private catalog.
- The private catalog specified in a STEPCAT DD statement in the step.
- The private catalog specified in a JOBCAT DD statement in the job, if the step does not contain a STEPCAT DD statement.

A private catalog can be either a VSAM user catalog or an integrated catalog facility (ICF).

**Use of Cataloging:** Cataloging allows you to keep track of the location of data sets. Cataloging also simplifies retrieving a data set: code only the DSNAMES parameter and OLD, SHR, or MOD in the DISP parameter and omit volume and device information.

**CATLG for a Cataloged Data Set:** Specify a disposition of CATLG for an already cataloged data set when adding to the data set if it may need another volume. The system updates the catalog entry to include the volume serial numbers of any additional volumes if the data set was specified as follows:

- DISP=(MOD,CATLG)
- No volume serial numbers were coded or referenced on the DD statement

**Generation Data Sets:** A collection of cataloged data sets that are kept in chronological order is 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. When creating a new generation data set, code a disposition of CATLG.

**When System Does Not Catalog a Data Set:** The system does not catalog a data set if the data set is not opened by the problem program and one of the following is true:

- The DD statement made a nonspecific request for a tape volume.
- The DD statement requested a tape volume for a tape device with dual density options but did not specify the density in the DEN subparameter of the DCB parameter.

## 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 delete the data set; only the reference in the catalog is removed. If you request the data set in a later job or step, the DD statement must specify volume information.

## Passing a Data Set

If more than one step in a job needs the same data set, each DD statement for the data set can pass it to a later step. A data set can be passed only within a job. A data set cannot be passed and received within the same step.

**To Pass:** To pass a data set, code PASS as the normal termination disposition; PASS cannot be the abnormal termination disposition. Code PASS each time the data set is needed until the last use in the job. In the last DD statement for the data set, assign it a final disposition.

**To Receive:** To receive a passed data set, specify in the DD statement the data set name without specifying a volume serial number or volume reference. Identical data set names, whether or not the names refer to the same data set, can be passed at the same time. Such identical data set names are received in the same order in which they are passed.

Do not try to receive a passed data set more times than it is passed.

**In a JES3 system,** if the data set was extended to additional volumes, code UNIT=AFF=ddname in the DD statement that receives the data set. This makes JES3 aware of the additional device needed for the extended data set.

**When Passing Step Abnormally Terminates:** If a step that passes a data set abnormally terminates during execution, the passed data set is passed. Thus, a following step that is executed because of a COND=EVENT or COND=ONLY can receive and process the passed data set. If the passed data set remains unreceived at the end of the job, the system performs the abnormal termination disposition, if specified, for the passed data set.

**Disposition Processing of Unreceived Passed Data Sets:** A job step can pass a data set that is never received by a later step. At the end of the job, the system processes the data set as an unreceived, passed data set.

**At Abnormal Termination when Abnormal Termination Disposition is Specified:** If a job step abnormally terminates, unreceived data sets that specified an abnormal termination disposition when passed are processed as specified in their abnormal termination dispositions.

For example, you code DISP=(,PASS,CATLG) for a new data set. If this step, or a later step before the receiving step, abnormally terminates during execution, the system tries to catalog the data set as instructed by the abnormal termination disposition of CATLG.

The following exceptions are not processed as specified in their abnormal termination dispositions. If the abnormal termination disposition requires an update to a private catalog and:

1. 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.
2. 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.
3. 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.

## End Processing

4. 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.

**At Abnormal Termination when No Abnormal Termination Disposition is Specified:** If no job step abnormally terminates before it begins execution, the system deletes all unreceived passed data sets that specified (NEW,PASS) and that did not specify an abnormal termination disposition; the system keeps all others. The system deletes those data sets even if they have unexpired expiration dates or retention periods.

**When Abnormal Termination Occurs Before Execution:** If a step abnormally terminates before it actually begins execution, for example, during allocation of devices and volumes or direct access space, the system ignores the disposition on the DD statement. The system keeps existing data sets and deletes new data sets.

**Deletion at End of Job:** If unreceived passed data sets are deleted at the end of a job, the system performs dynamic allocation to allocate a device and volume for deletion. Depending on the JOB statement MSGLEVEL parameter or the installation defaults, the system issues allocation messages for these data sets.

### Default Disposition Processing

If you omit the DISP parameter or one of its subparameters, the system supplies default values.

If the data set status is omitted, the system assumes NEW. If the second or third subparameter is omitted, 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. The system treats a data set as existing when the status is OLD, SHR, or MOD with volume information.
- Data sets created in the job are automatically deleted. The system treats a data set as newly created when the status is NEW, omitted, or MOD without volume information.

### 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.

## Examples

---

```
//DISPJ JOB 158765,'SECT. 27'
//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,CATLG,KEEP)
//D4 DD DSN=&&SYS1,DISP=(MOD,PASS),UNIT=3350,
// VOL=SER=335004,SPACE=(TRK,(15,5,1))
//S2 EXEC PGM=IEFBR14
//D5 DD DSN=&&SYS1,DISP=(MOD,DELETE),UNIT=3350,
// VOL=SER=335004,SPACE=(TRK,(15,5,1))
```

1. D1 requests a data set that already exists and can be shared with other jobs. It is to be kept on the volume at the end of step S1.
2. D2 requests a data set that already exists and cannot be shared with other jobs. It is to be deleted at the end of S1, but is to be kept and uncataloged if S1 abnormally terminates.
3. D3 defines a new data set that is to be assigned to volume 335001 on a 3350 Direct Access Storage device. The data set is to be kept on the volume and cataloged if S1 terminates normally, but is to be kept and not cataloged if S1 terminates abnormally.
4. D4 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 and allocated 15 primary tracks, five secondary tracks, and one directory record. This data set is to be passed for use in a later step in this job.
5. D5 requests the temporary data set passed by D4 of S1. When S2 completes, the data set is to be deleted.

---

```
//PASS JOB , 'BILL H.'
//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; DD8 receives the data set passed by DD4. DD8 continues to pass the data set originally passed by DD4.
  3. DD9 receives the data set passed by DD8.
-

## End Processing

---

Cataloged procedure MYPROC:

```
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSNNAME=&A,DISP=(NEW,PASS),
// SPACE=(TRK,(1,1)),UNIT=SYSDA
//DD2 DD DSNNAME=*.DD1,DISP=(OLD,PASS),
// VOL=REF=*.DD1
//STEP2 EXEC PGM=IEFBR14
//DD3 DD DSNNAME=&A,DISP=(OLD,DELETE)
```

Input stream:

```
//JOBEX JOB
//S1 EXEC PROC=MYPROC
//S2 EXEC PROC=MYPROC
```

A problem can occur when the same data set is passed more times than it is received in a procedure that is called more than once in a job.

DD1 and DD2 pass data set &A. DD3 receives data set &A. After the procedure has been executed, one entry for data set &A remains unreceived.

When the procedure is called a second time, DD3 receives data set &A from the first execution of the procedure. 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.

---

## Disposition Controlled by Time

When creating a data set, tell the system how long to keep it by coding:

```
//ddname DD LABEL=RETPD=nnnn,...
//ddname DD LABEL=EXPDT=yyddd,...
```

As long as the time period has not expired, the system will not delete or write over a data set on direct access space. This is true even if a DD statement specifies a disposition of DELETE for the data set; the system will not delete a data set until after the expiration date or retention period.

When the expiration date of a data set is the current date, the data set is considered expired. The system will delete it, if requested in a DD statement, or write over it.

**Deleting before Expiration Date or Retention Period:** If it is necessary to delete a data set before the expiration date or retention period, do one of the following:

- 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 *Integrated Catalog Administration: Access Method Services Reference* or *VSAM Catalog Administration: Access Method Services Reference*.
- For data sets cataloged in a non-VSAM catalog, delete the catalog entry with the IEHPROGM utility as described in *Data Administration: Utilities*.



- For the data set control block, use a SCRATCH macro with the OVRD parameter; this makes the space occupied by that data set available for reallocation. See *System-Data Administration*.

### Examples

---

```
//D1 DD DSN=MYDS,DISP=(NEW,CATLG,DELETE),UNIT=DISK,  
// VOLUME=SER=223344,LABEL=(,SL,RETPD=365)  
//D2 DD DSN=DSABC,DISP=(NEW,KEEP),UNIT=TAPE,  
// VOLUME=SER=158716,LABEL=(,SUL,EXPDT=90032)
```

---

## Release of Unused Direct Access Space in End Processing

To request that the system release direct access space that was allocated to an output data set but was not used, code:

```
//ddname DD SPACE=(TRK,(quantity),RLSE),...  
//ddname DD SPACE=(CYL,(quantity),RLSE),...  
//ddname DD SPACE=(blklgth,(quantity),RLSE),...
```

The system releases space only if the data set is open for output and the last operation was a write. The system does not release space if the step terminates abnormally. The system ignores a request to release unused space if:

- Another job is sharing the data set.
- Another task in the same job is processing an OPEN, CLOSE, EOVS, or FEOVS request for the data set.
- Another data control block is open for the data set.
- The CLOSE macro instruction contains TYPE=T.

### Example

---

```
//DD3 DD DSN=DEPTDS,DISP=(NEW,KEEP),UNIT=DISK,  
// SPACE=(CYL,20,RLSE)
```

---

## End Processing

### Disposition End Processing of Volume

Disposition of the tape or direct access volume containing a data set is controlled by coding:

```
//ddname DD VOLUME=(PRIVATE,RETAIN,...),...  
//ddname DD VOLUME=(PRIVATE,...),...  
//ddname DD VOLUME=(,RETAIN,...),...
```

**RETAIN Support:** RETAIN can be specified only for tape.

In a JES3 system, RETAIN is supported only by MVS. If coded on a DD statement for a data set on an MVS-managed tape device, the system designates the volume as retained. If coded on a DD statement for a data set on a JES3-managed device, it is ignored.

#### Disposition of Removable Volumes

If a removable direct access or tape volume is designated as private, the system asks the operator to demount the volume at the end of the step and place it in the installation library.

If a removable direct access or tape volume is designated as public, the system keeps it mounted for other uses, unless the device is needed for another allocation.

*Note:* Mass storage volumes, while ultimately recorded on tape-like media, are treated like direct access.

**Tape Volumes in JES2:** When disposing of tape volumes, a JES2 system marks them as follows:

- **Keep (K):** The volume is to be placed in the installation tape library. K is the designation for a private tape volume.
- **Scratch (D):** The volume can be used whenever a DD statement makes a nonspecific request for a tape volume. D is the designation for a public tape volume.

#### Examples

---

Volumes treated as private, demounted, and kept:

```
//EX1 DD DSNAME=A,DISP=(NEW,KEEP),VOLUME=PRIVATE,UNIT=TAPE  
//EX2 DD DSNAME=B,DISP=OLD,VOLUME=SER=223344,UNIT=DISK  
//EX3 DD DSNAME=H,DISP=OLD
```

Volumes treated as public and kept mounted for other uses:

```
//EX4 DD DSNAME=D,UNIT=TAPE  
//EX5 DD DSNAME=##TEMP,UNIT=DISK
```

---

## Volume Retention

The system designates a tape volume as *retained* (R) if the volume contains one of the following:

- A passed data set
- A data set requested by a DD statement with RETAIN in the VOLUME parameter.

**Retained Private Tape Volume:** If RETAIN is coded or the data set is passed, the system designates the volume as R, keeps the volume mounted, and does not rewind the tape when the data set is closed or at the end of the step.

**Retained Public Tape Volume:** If RETAIN is coded or the data set is passed, the system designates the volume as R, but asks the operator to demount it and keep it near for possible use later.

**Use of Retained Volumes:** In a multiple step job, if there is a period when a volume is not in use, you can specify RETAIN to try to keep the volume mounted. If the volume remains mounted, the operator does not have to demount and remount it, and the job does not have to wait until the volume is remounted.

**Demounting of Passed or Retained Volumes:** Even if you specify RETAIN or a disposition of PASS, the operator can still unload the volume or, if the device is needed for another step in the same or another job, the system can allocate the device and demount the volume. Either can occur when the device on which the volume is mounted is not allocated to the job step that specified RETAIN or, for unlabeled tapes, when the volume requires verification.

### Example

---

```
//EXDD DD DSNAME=TAPEDS,DISP=(NEW,CATLG,DELETE),UNIT=3420,  
// VOLUME=(PRIVATE,RETAIN)
```

---



## Part 5. Tasks for Requesting Sysout Data Set Resources

This part describes how to create system output (sysout) data sets, which are output data sets processed by JES2 or JES3. The task required to request a sysout data set is:

- Identification

Other tasks can optionally be performed:

- Description
- Performance control
- Processing control
- End processing
- Output destination
- Output formatting
- Output limiting

**Processing Output:** The two ways to process output data sets are:

- Define a sysout data set and how it is to be processed and allow the job entry subsystem to process it. JES writes the data set to a spool device. Then JES or an external writer prints or punches it on a local or remote printer or punch, or JES transmits it to a remote output device or node.
- Define an output data set and specify in the DD statement UNIT parameter the device on which the output should be written. The system allocates the device exclusively to the job. Data management routines write the output from the program to the specified device.

This part describes how sysout data sets are defined and processed.

# Part 5

## Part 5 Contents

<b>Chapter 18. Identification</b>	18-1
Identification as a Sysout Data Set	18-1
Examples	18-1
Identification of Output Class	18-2
Examples	18-2
Identification of Data Set on 3540 Diskette Input/Output Unit	18-2
Example	18-3
<b>Chapter 19. Description</b>	19-1
Description of Data Attributes	19-1
Example	19-1
<b>Chapter 20. Performance Control</b>	20-1
Performance Control by Queue Selection	20-1
Ignoring Priority	20-1
Example	20-1
<b>Chapter 21. Processing Control</b>	21-1
Processing Control with Additional Parameters	21-2
Adding Parameters from OUTPUT JCL Statement	21-2
Multiple References	21-2
Examples	21-2
Adding Parameters from JES2 /*OUTPUT Statement	21-4
Adding Parameters from JES3 /**FORMAT Statement	21-4
Processing Control with Other Data Sets	21-5
Using Output Class	21-5
Examples	21-5
Using Sysout Data Set Size in a JES3 System	21-5
Use of THRESHLD	21-5
Examples	21-6
Using Groups in a JES2 System	21-6
Subgroups	21-6
Demand Setup Groups	21-6
Example	21-6
Processing Control by External Writer	21-7
Examples	21-7
Processing Control by Mode	21-7
Examples	21-7
Processing Control by Holding	21-8
Uses for Holding	21-8
Releasing Held Data Set	21-8
Printing Released Data Set	21-8
Examples	21-9
Processing Control by Suppressing Output	21-9
Using Dummy Status to Suppress Output	21-9
Effect of Dummy Sysout Data Set	21-9
Use of a Dummy Sysout Data Set	21-9
Nullifying a Dummy Sysout Data Set	21-9
Examples	21-9
Using Class to Suppress Output in a JES2 System	21-10
Use of Output Suppression	21-10

Examples	21-10
Processing Control with Checkpointing	21-10
Examples	21-11
Processing Control by Print Services Facility	21-11
Examples	21-11
<b>Chapter 22. End Processing</b>	22-1
Deallocation End Processing	22-1
Spinning off Data Sets	22-1
Use of Spinning Off	22-1
Example	22-1
<b>Chapter 23. Destination Control</b>	23-1
Destination Control to Local or Remote Device or to Another Node	23-1
Multiple Destinations	23-2
Controlling Output Destination in a JES2 Network	23-2
Examples	23-3
Controlling Output Destination in a JES3 Network	23-3
Output Destination when Remote Job Processing in JES3	23-3
Examples	23-3
Destination Control to Another Processor in a JES3 System	23-4
Example	23-4
Destination Control to Internal Reader	23-4
Message Class for Internal Reader Job	23-4
Limiting Records to Internal Reader	23-4
Sending Internal Reader Buffer Directly to JES	23-4
References	23-5
Example	23-5
Destination Control to Terminal	23-6
Example	23-6
<b>Chapter 24. Output Formatting</b>	24-1
Output Formatting to Any Printer	24-2
3203 Printer Model 5 in a JES2 System	24-2
Examples	24-2
Output Formatting to 3800 Printing Subsystem	24-3
Copy Modification	24-3
Character Arrangements	24-3
Modifying Character-Arrangement Tables	24-3
Dynamically Selecting Character-Arrangement Tables	24-4
When Data Set Printed on 3800 or Other Printers	24-4
Examples	24-4
Output Formatting to 3211 Printer with Indexing Feature in a JES2 System	24-4
Examples	24-5
Output Formatting to Punch	24-5
Interpretation of Punched Cards	24-5
Interpretation in a JES3 System	24-5
Examples	24-5
Output Formatting of Dumps on 3800 Printing Subsystem	24-6
Examples	24-6
<b>Chapter 25. Output Limiting</b>	25-1
Output Limiting	25-1
Use of Limiting	25-1

## **Part 5**

<b>Actions when Limit Exceeded in a JES3 System</b>	<b>25-2</b>
<b>Examples</b>	<b>25-2</b>



## Chapter 18. Identification

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Identification					
as a sysout data set	SYSOUT				
of output class	class on SYSOUT	CLASS	MSGCLASS on JOB with SYSOUT=* or CLASS=* and SYSOUT=(,)		
of data set on 3540 Diskette Input/Output Unit	DSID				

Figure 18-1. Identification Task for Requesting Sysout Data Set Resources

### Identification as a Sysout Data Set

To define an output data set as a sysout data set, code:

```
//ddname DD SYSOUT=class
//ddname DD SYSOUT=(class,writer-name,form-name)
//ddname DD SYSOUT=(class,writer-name,code-name)
//ddname DD SYSOUT=*
//ddname DD SYSOUT=(,)
```

#### Examples

---

```
//EX1 DD SYSOUT=B
//EX2 DD SYSOUT=(A,,FM23)
//EX3 DD SYSOUT=(F,,CD3)
//EX4 DD SYSOUT=*

//EX5 OUTPUT CLASS=E
//EX6 DD SYSOUT=(,),OUTPUT=*,EX5
```

---

## Identification

### Identification of Output Class

The installation sets up output classes during JES2 or JES3 initialization. Each class is assigned processing characteristics and is printed or punched on certain devices. The output class for a sysout data set is identified by coding one of the following:

```
//ddname DD SYSOUT=class

//jobname JOB acct,progname,MSGCLASS=class
//stepname EXEC PGM=x
//ddname DD SYSOUT=*

//name OUTPUT CLASS=class
//ddname DD SYSOUT=(,),OUTPUT=*.name
```

For example, the installation could define output class W to contain low-priority output; class Y to contain output to be printed on a special form, so that the JCL would not need to request the form; and class J to be reserved for high-volume output.

To print the sysout data set and messages from the job on the same output listing, see "Printing Job Log and Sysout Data Sets Together" on page 7-7.

#### Examples

---

```
//X1 DD SYSOUT=A

//JOBA JOB , 'C. SARDO',MSGCLASS=B
//ST1 EXEC PGM=ANY
//X2 DD SYSOUT=*

//OUTA OUTPUT CLASS=C
//X3 DD SYSOUT=(,),OUTPUT=*.OUTA
```

---

### Identification of Data Set on 3540 Diskette Input/Output Unit

Data sets are written on 3540 diskette volumes by coding:

```
//ddname DD SYSOUT=(class,diskette-writer),DSID=id
//ddname DD SYSOUT=(class,diskette-writer),DSID=(id,V)
```

A system command, from the operator or in the input stream, must start the diskette writer before the DD statement is processed.

For more information on the 3540 diskette, see *IBM 3540 Programmer's Reference*. For information on external writers, see *SPL: System Modifications*.

## Example

---

```
//EX7 DD SYSOUT=(W,WRT3540),DSID=MYDS5
```

---



## Chapter 19. Description

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Description					
of data attributes	DCB				

Figure 19-1. Description Task for Requesting Sysout Data Set Resources

### Description of Data Attributes

When JES2 or JES3 processes the sysout data set, omit the DCB parameter. If an external writer processes the sysout data set, code a DCB parameter if required to complete data control block fields in the writer.

#### Example

---

```
//OUT3 DD SYSOUT=(H,WRTPGM),DCB=(RECFM=FB,LRECL=133,BLKSIZE=532)
```

---



## Chapter 20. Performance Control

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Performance control					
by queue selection		PRTY			

Figure 20-1. Performance Control Task for Requesting Sysout Data Set Resources

### Performance Control by Queue Selection

You can specify the priority at which the sysout data set enters the output queue by coding:

```
//name OUTPUT PRTY=nnn
```

Use the priority to increase a sysout data set's priority so it will be printed sooner than it otherwise might have been.

**Ignoring Priority:** The installation can instruct the system to ignore a priority specified on an OUTPUT JCL statement.

#### Example

---

```
//OUTA OUTPUT PRTY=255
//MYDS DD      SYSOUT=F,OUTPUT=*.OUTA
```

This example requests the highest priority possible.

---





## Chapter 21. Processing Control

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Processing control					
with additional parameters	OUTPUT code-name on SYSOUT	DEFAULT			
with other data sets	class on SYSOUT	THRESHLD (JES3 only) GROUPID (JES2 only)			
by external writer	writer-name on SYSOUT	WRITER			
by mode		PRMODE			
by holding	HOLD class on SYSOUT				
by suppressing out- put	DUMMY class on SYSOUT				
with checkpointing		CKPTLINE CKPTPAGE CKPTSEC CKPTSEC			
by Print Services Facility (PSF)		FORMDEF PAGEDEF			

Figure 21-1. Processing Control Task for Requesting Sysout Data Set Resources

## Processing Control

### Processing Control with Additional Parameters

To control how a sysout data set is processed, specify parameters on the DD statement with the SYSOUT parameter. Code the following statements to supply additional parameters:

By explicit reference to earlier OUTPUT JCL statement:  
//name OUTPUT parameters  
//ddname DD SYSOUT=class,OUTPUT=\*.name,parameters

By implicit reference to earlier default OUTPUT JCL statement:  
//name OUTPUT DEFAULT=YES,parameters  
//ddname DD SYSOUT=class,parameters

#### Adding Parameters from OUTPUT JCL Statement

JES combines the parameters from the sysout DD statement and one OUTPUT JCL to write the sysout data set. If a parameter appears on both statements, JES uses the parameter from the DD statement.

**Multiple References:** A sysout DD statement can reference more than one OUTPUT JCL statement. For each reference to an OUTPUT JCL statement, JES processes the sysout data set using the parameters of the DD statement combined with the parameters from one of the OUTPUT JCL statements.

#### Examples

---

```
//JOB1 JOB , 'DEPT. 25'  
//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
```

This example shows an explicit reference to an OUTPUT JCL statement. Note that with an explicit reference, all default OUTPUT JCL statements are ignored.

- 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).
-

```

//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 )
//

```

This example shows how the position of the OUTPUT JCL statement affects the processing of the sysout data sets.

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.

## Processing Control

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.

---

```
//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.

---

### Adding Parameters from JES2 /\*OUTPUT Statement

JES2 can combine the parameters from the sysout DD statement and a referenced /\*OUTPUT statement to write the sysout data set.

Because the OUTPUT JCL statement provides greater output processing capabilities, an installation should consider changing its /\*OUTPUT statements to OUTPUT JCL statements.

Be careful when doing the change. Before the change, the third subparameter in the DD SYSOUT parameter references a JES2 /\*OUTPUT statement. But, if the DD statement references an OUTPUT JCL statement, the system interprets the third subparameter as the name of forms to be used in processing the sysout data set.

### Adding Parameters from JES3 /\*FORMAT Statement

A JES3 /\*FORMAT statement can explicitly reference a sysout DD statement to make JES3 combine the parameters from the sysout DD statement and the /\*FORMAT statement to write the sysout data set.

Because the OUTPUT JCL statement provides greater output processing capabilities, an installation should consider changing its /\*FORMAT statements to OUTPUT JCL statements.

## Processing Control with Other Data Sets

### Using Output Class

JES prints on the same output listing the output from all sysout data sets for a job if the class, forms, FCB, UCS, and DEST parameters are the same and if an external writer is not specified. The installation can choose to print all sysout data sets that specify the same output class as the JOB statement MSGCLASS parameter on the same listing, even though the forms, FCB, UCS, and sometimes the DEST parameters are different.

#### *Examples*

---

```
//DD1  DD  SYSOUT=(C,,FM34)
//DD2  DD  SYSOUT=(C,,FM34)
```

The sysout data sets for DD1 and DD2 are written on the same output listing.

---

```
//JEX  JOB  , 'M. BIRDSALL',MSGCLASS=D
//ST1  EXEC  PGM=WKRPT
//DDA  DD  SYSOUT=*
//DDB  DD  SYSOUT=D
```

The sysout data sets for DDA and DDB are written on the same output listing as the job log.

---

### Using Sysout Data Set Size in a JES3 System

To control whether all the sysout data sets in one class from a job are printed together or as separate units of work, code one of the following groups:

```
//name  OUTPUT THRESHLD=limit
//ddname1 DD  SYSOUT=class,OUTPUT=*.name
//ddname2 DD  SYSOUT=class,OUTPUT=*.name

//name  OUTPUT DEFAULT=YES,CLASS=class,THRESHLD=limit
//ddname1 DD  SYSOUT=(,)
//ddname2 DD  SYSOUT=(,)
```

JES3 calculates the size of the sysout data set(s) as the number of records multiplied by the number of copies requested. When the size exceeds the THRESHLD value, JES3 creates a new unit of work on a data set boundary, and queues it for printing.

**Use of THRESHLD:** If a sysout data set or all the sysout data sets in the same class from a job are large or large numbers of copies are requested, the THRESHLD limit can be used to print copies simultaneously by different printers.

# Processing Control

## Examples

---

```
//OUTA  OUTPUT THRESHLD=10000
//MYDS1 DD      SYSOUT=C,OUTPUT=*.OUTA,COPIES=5
//GRDS  DD      SYSOUT=C,OUTPUT=*.OUTA,COPIES=3

//OUTB  OUTPUT DEFAULT=YES,CLASS=C,THRESHLD=10000
//MYDS1 DD      SYSOUT=(,),COPIES=5
//GRDS  DD      SYSOUT=(,),COPIES=3
```

---

## Using Groups in a JES2 System

In JES2 systems, you can group sysout data sets together by coding:

```
//name  OUTPUT GROUPID=output-group
```

Sysout data sets in the same group are processed together in the same location and time.

**Subgroups:** You can always group sysout data sets with similar processing characteristics. But, you cannot group sysout data sets with different output classes, destinations, processing modes (PRMODE), writer names, or groupids. If you use GROUPID to group dissimilar data set, the system breaks down the group into subgroups of sysout data sets with identical classes, destinations, processing modes, writer names, and groupids.

**Demand Setup Groups:** The installation controls if a group can contain sysout data sets with different printer setup requirements, such as forms. Such groups are called demand setup groups. If demand setup grouping is not permitted, data sets with different setup requirements are placed in different subgroups.

## Example

---

```
//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 subgroups are created for the three sysout data sets because of the different output classes. One subgroup contains data sets RP1 and RP3; the other contains RP2.

---

## Processing Control by External Writer

To request that a sysout data set be processed by an IBM-supplied or user-written external writer, rather than the installation's JES, code one of the following:

```
//ddname DD SYSOUT=(class,writer-name)

//name OUTPUT WRITER=writer-name
//ddname DD SYSOUT=class,OUTPUT=*.name

//name OUTPUT DEFAULT=YES,WRITER=writer-name
//ddname DD SYSOUT=class
```

For an external writer, the operator determines which sysout data sets are selected. This can cause certain data sets to be printed on the same listing even though all of the forms, FCB, UCS, and DEST parameters are not the same. The operator must start the external writer for a sysout data set to be printed or punched.

For more information on external writers, see *System Modifications*.

### Examples

---

```
//DS1 DD SYSOUT=(H,MYWRIT)

//OTA OUTPUT WRITER=MYWRIT
//DS1 DD SYSOUT=H,OUTPUT=*.OTA

//OTB OUTPUT DEFAULT=YES,WRITER=MYWRIT
//DS1 DD SYSOUT=H
```

---

## Processing Control by Mode

To request the correct process mode for a sysout data set, code one of the following:

```
//name OUTPUT PRMODE=LINE
//name OUTPUT PRMODE=PAGE
//name OUTPUT PRMODE=process-mode
```

JES schedules the sysout data set to a printer that can operate in the specified mode.

### Examples

---

```
//OTS OUTPUT PRMODE=PAGE
//ABC DD SYSOUT=F,OUTPUT=*.OTS
```

---

JES schedules data set ABC to a 3800 Printing Subsystem Model 3, which can print in page mode. Output class F must handle processing for a 3800 model 3.

---

## Processing Control

### Processing Control by Holding

To hold a sysout data set on the JES spool and delay its printing or punching, code one of the following:

```
//ddname DD SYSOUT=class,HOLD=YES
```

Where the specified class is designated as a held class during JES initialization:

```
//ddname DD SYSOUT=class
```

```
//name OUTPUT CLASS=class  
//ddname DD SYSOUT=(,),OUTPUT=*.name
```

```
//name OUTPUT DEFAULT=YES,CLASS=class  
//ddname DD SYSOUT=(,)
```

#### Uses for Holding

Some of the reasons for holding a data set are:

- To make it available for inspection from a time-sharing terminal.
- If it is very large, to prevent it from monopolizing an output device until smaller data sets are written.
- If it requires special forms, to delay its printing or punching until the operator can supply the forms.

#### Releasing Held Data Set

When a data set is to be held, JES places the sysout data set on a hold queue until the operator releases it. The system issues no message to tell the operator that the data set is being held. Therefore, when the data set can be processed, ask the operator to release it or release it from a TSO userid with a TSO OUTPUT command. See *TSO Command Language Reference* for information on TSO commands.

**Printing Released Data Set:** In a JES2 system, 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. A spun-off data set is made available for printing or punching before the job is completed; spun-off data sets are always printed separately.
- JES has not started printing the job's system-managed data sets.
- JES is not printing multiple copies of the job's system-managed data sets.
- If the released data sets had not been held, they would have been printed with the job's system-managed data sets.

If all of these criteria are not satisfied, the released data sets are printed separately from the job's system-managed data sets.



## Examples

---

```
//DD1 DD SYSOUT=C,HOLD=YES
//DD2 DD SYSOUT=J
//OT1 OUTPUT CLASS=J
//DD3 DD SYSOUT=(,),OUTPUT=*.OT1
//OT2 OUTPUT DEFAULT=YES,CLASS=J
//DD4 DD SYSOUT=(,)
```

In all these examples, the installation defined class J as a held class during JES initialization.

---

## Processing Control by Suppressing Output

### Using Dummy Status to Suppress Output

If you want to suppress processing of a sysout data set, assign it a dummy status by coding:

```
//ddname DD DUMMY,SYSOUT=class,...
```

**Effect of Dummy Sysout Data Set:** When DUMMY is coded, the system ignores the SYSOUT parameter and bypasses all output operations to the spool. The sysout data set is not printed or punched.

**Use of a Dummy Sysout Data Set:** Defining a sysout data set as a dummy data set is useful when testing a program; you do not want data sets printed until you are sure they contain meaningful output.

**Nullifying a Dummy Sysout Data Set:** When the sysout data set is to be processed, remove the DUMMY parameter from the sysout DD statement.

## Examples

---

```
//EXA DD DUMMY,SYSOUT=A
//EXB DD DUMMY,SYSOUT=(B,WRT),DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
```

---

## Processing Control

### Using Class to Suppress Output in a JES2 System

To suppress the printing or punching of a sysout data set in a JES2 system, code one of the following:

```
//ddname DD SYSOUT=class

//name OUTPUT CLASS=class
//ddname DD SYSOUT=(,),OUTPUT=*.name

//name OUTPUT DEFAULT=YES,CLASS=class
//ddname DD SYSOUT=(,)
```

During JES2 initialization, the installation must specify that the requested class contains data sets that are deleted before being printed or punched.

*Use of Output Suppression:* Use this technique to suppress the output of started tasks.

#### Examples

---

```
//DD2 DD SYSOUT=S

//OT1 OUTPUT CLASS=S
//DD3 DD SYSOUT=(,),OUTPUT=*.OT1

//OT2 OUTPUT DEFAULT=YES,CLASS=S
//DD4 DD SYSOUT=(,)
```

In all these examples, the installation defined class S as an output suppression class.

---

## Processing Control with Checkpointing

To write a checkpoint while JES is processing a sysout data set, code one of the following:

In a JES2 system, for any device and, in a JES3 system, when Print Services Facility (PSF) prints the data set on a 3800 Printing Subsystem Models 3 and 8:

```
//SYSCKEOV DD parameters

//name OUTPUT CKPTLINE=number,CKPTPAGE=number
//ddname DD SYSOUT=class

//SYSCKEOV DD parameters

//name OUTPUT CKPTSEC=number
//ddname DD SYSOUT=class
```

## Examples

---

```
//J2      JOB      ,MHB
//SYSCKEOV DD      DSNAME=CK,UNIT=TAPE,DISP=(NEW,DELETE,KEEP)
//S1      EXEC     PGM=ABC
//OT2     OUTPUT   CKPTLINE=60,CKPTPAGE=40
//DDB     DD       SYSOUT=C
```

JES2 writes a checkpoint in the SYSCKEOV data set every 40 logical pages. A logical page contains 60 lines.

---

```
//J2      JOB      ,MHB
//SYSCKEOV DD      DSNAME=CK,UNIT=TAPE,DISP=(NEW,DELETE,KEEP)
//S1      EXEC     PGM=ABC
//OT2     OUTPUT   CKPTSEC=60
//DDB     DD       SYSOUT=C
```

JES2 writes a checkpoint in the SYSCKEOV data set every 60 seconds.

---

## Processing Control by Print Services Facility

To control how the Print Services Facility (PSF) prints a sysout data set on a 3800 Printing Subsystem Model 3, code:

```
//name    OUTPUT  PAGEDEF=membername,FORMDEF=membername
//ddname  DD      SYSOUT=class,OUTPUT=*.name
```

The PAGEDEF and FORMDEF parameters identify library members in SYS1.IMAGELIB. These members contain statements that specify how PSF is to process the sysout data set.

## Examples

---

```
//OTPSF  OUTPUT  PAGEDEF=PSLONG,FORMDEF=FSBILL
//MYPNT  DD      SYSOUT=N,OUTPUT=*.OTPSF
```

---



## Chapter 22. End Processing

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
End processing					
deallocation	FREE				

Figure 22-1. End Processing Task for Requesting Sysout Data Set Resources

### Deallocation End Processing

Normally JES2 or JES3 schedules all sysout data sets from a job for printing or punching when all the system-managed data sets are processed at the end of the job.

**Spinning off Data Sets:** Sysout data sets can be scheduled for printing or punching when the data set is closed before the job completes execution. Code:

```
//ddname DD SYSOUT=class,FREE=CLOSE
```

These data sets are called **spun-off data sets**.

If the step continues processing after the close, the sysout data set may be printed concurrently with the last of the step's execution.

**Use of Spinning Off:** Use FREE=CLOSE to let JES begin printing or punching a sysout data set before a long job step is finished.

#### Example

---

```
//STEP1 EXEC PGM=VERYLONG
//SHORT DD SYSOUT=C,FREE=CLOSE
```

---



## Chapter 23. Destination Control

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Destination control</b>					
to local or remote device or to another node	DEST class on SYSOUT	DEST COMPACT		/*ROUTE PRINT /*ROUTE PUNCH	ORG on /*MAIN
to another processor					ACMAIN on /*MAIN
to internal reader	INTRDR as writer-name on SYSOUT		/*EOF /*DEL /*PURGE /*SCAN		
to terminal	TERM				

Figure 23-1. Destination Control Task for Requesting Sysout Data Set Resources

### Destination Control to Local or Remote Device or to Another Node

To send a sysout data set to a local or remote device or to another node, code one of the following:

```

//ddname DD      SYSOUT=class,DEST=destination
//name   OUTPUT  DEST=destination,COMPACT=compaction-table-name
//ddname DD      SYSOUT=class,OUTPUT=*.name

```

In a JES2 system:

```

/*ROUTE PRINT destination
//ddname DD      SYSOUT=class

/*ROUTE PUNCH destination
//ddname DD      SYSOUT=class

```

To send to group, node, or remote work station in a JES3 system:

```

//jobname JOB acct,progname
/*MAIN ORG=group-or-node.remote
//stepname EXEC PGM=x
//ddname DD      SYSOUT=class

```

# Destination Control

## Multiple Destinations

For example, to print a report in Chicago, New York, Paris, and Los Angeles, code and reference four OUTPUT JCL statements. Specify a different destination on each; you can code only one destination on each OUTPUT JCL statement.

By referencing OUTPUT JCL statements, you can specify 128 different destinations for a single sysout data set. In addition, you can use each OUTPUT JCL statement to specify processing options for each destination.

## Controlling Output Destination in a JES2 Network

In a network, you can route sysout data sets from any node or work station to any node or work station.

Unless overridden by the operator or directed by a destination parameter, a sysout data set is printed or punched at the submitting location. To route a sysout data set to another location, use the following:

### **DEST parameter on DD SYSOUT statement**

Specifies the destination for the sysout data set being defined.

### **class subparameter in SYSOUT parameter on DD statement**

Specifies the destination for the sysout data set being defined. During JES2 initialization, a destination must have been defined for the requested class.

### **DEST parameter on OUTPUT JCL statement**

Specifies the destination for all referencing sysout data sets.

### **DEST parameter on /\*ROUTE PRINT or PUNCH statement**

Specifies the destination of a job's sysout data sets for any node or any remote work station. All sysout data sets that have no specific destination go to the destination in the /\*ROUTE statement.

*Note:* If you send a job to execute at a remote node and the job has a ROUTE PRINT RMTnnn statement, JES2 returns the output to RMTnnn at the node of origin. For JES2 to print the output at RMTnnn at the executing node, code DEST=NnnnRmmm on an OUTPUT JCL statement or sysout DD statement.

### **Default Output Destination**

Implicitly specifies the destination. When the job enters the job entry network, the default destination is determined by the device from which the job entered the system.



## Examples

---

```
//DDFAR1 DD      SYSOUT=E,DEST=NYC
//DDFAR2 DD      SYSOUT=F
//OTFAR  OUTPUT  DEST=NYC,COMPACT=TABCM
//DD1    DD      SYSOUT=E,OUTPUT=*.OTFAR
/*ROUTE  PRINT   NYC
//DD3    DD      SYSOUT=E
/*ROUTE  PUNCH   NYC
//DD4    DD      SYSOUT=P
```

For the second example, output class F must be defined during JES2 initialization as having a destination, for example, a node in Los Angeles.

---

## Controlling Output Destination in a JES3 Network

In a network, you can route sysout data sets from any node or work station to any node or work station.

Unless changed by the **ORG** parameter on a **//\*MAIN** statement or directed by a destination parameter, a sysout data set is printed or punched at the submitting location. To route a sysout data set to another location, use the following:

### **DEST** parameter on **DD SYSOUT** Statement

Specifies the destination for the sysout data set being defined.

### **DEST** parameter on **OUTPUT JCL** Statement

Specifies the destination for all referencing sysout data sets.

### **ORG** parameter on **//\*MAIN** Statement

Specifies an origin group, network node, or remote work station for the job's sysout data sets.

**Output Destination when Remote Job Processing in JES3:** For jobs from remote work stations submitted through remote job processing (RJP), the sysout data sets are returned to the originating work station unless another destination is requested in a **//\*MAIN** statement with an **ORG** parameter, **OUTPUT JCL** statement, or **DD** statement.

## Examples

---

```
//DDFAR  DD      SYSOUT=E,DEST=NYC
//OTFAR  OUTPUT  DEST=NYC,COMPACT=TABCM
//DD1    DD      SYSOUT=E,OUTPUT=*.OTFAR
//JEX3   JOB     , 'MAIL A60'
/*MAIN   ORG=NYC
//S3     EXEC    PGM=GHI
//DD4    DD      SYSOUT=E
```

---

## Destination Control

### Destination Control to Another Processor in a JES3 System

To direct all of a job's sysout data sets to a TSO userid on another processor, code:

```
//*MAIN ACMAIN=processor-id,USER=userid
```

#### Example

---

```
//J1 JOB ,MHB  
//*MAIN ACMAIN=2,USER=D17MHB  
//S1 EXEC PGM=PROG67  
//DDA DD SYSOUT=G
```

---

### Destination Control to Internal Reader

To make a sysout data set from a job or step the input to another job or step, direct the data set to the internal reader. The input to the internal reader must be the JCL statements to run the later job. Code:

```
//ddname DD SYSOUT=(class,INTRDR)
```

INTRDR is an IBM-reserved name identifying the internal reader as the program to process the sysout data set. 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 by JES and used as input to the job you specify.

**Message Class for Internal Reader Job:** The output class in the SYSOUT parameter becomes the default message class for the job going into the internal reader, unless you code the MSGCLASS parameter on the JOB statement.

**Limiting Records to Internal Reader:** Use the OUTLIM parameter on the DD statement to limit the number of logical records written to the internal reader.

**Sending Internal Reader Buffer Directly to JES:** Instead of waiting for the buffer in your address space to fill up, send the contents of the internal reader buffer directly to JES by coding as the last record in the job:

**/\*EOF**

For JES2, this control statement delimits the job in the data set 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 closes the data set and sends the data set to the JES3 input service. Because the internal reader closes the data set without deallocating it, it is available for more records.

## **/\*DEL**

For JES2, this control statement cancels the job in the data set and schedules it for immediate output processing. The output consists 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**

For JES2 only, this control statement cancels the job in the data set and schedules it for purge processing; no output is produced for the job.

## **/\*SCAN**

For JES2 only, this control statement requests that JES2 only scan the job in the data set for JCL errors. The job is not to be executed.

**References:** For more information on the internal reader, see *System Modifications, SPL: JES2 Initialization, and Tuning*, or *JES3 SPL: Initialization and Tuning*.

### **Example**

---

```

//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 JOBB and JOBC from in-stream data set SYSUT1 and writes them to sysout data set SYSUT2, which is submitted to the internal reader.
  - The message class for JOBB and JOBC is M, the SYSOUT class specified on DD statement SYSUT2.
  - The message class for sysout data set OUTDD1 is M because SYSOUT=\* is coded.
  - The /\*EOF statement specifies that the preceding jobs are to be sent immediately to JES for input processing.
-

## Destination Control

### Destination Control to Terminal

To indicate that a sysout data set is going to a terminal for a TSO user, code:

```
//ddname DD TERM=TS
```

In a batch job, TERM=TS is treated as though SYSOUT=\* were coded. For an output data set in a foreground job, TERM=TS specifies that the data set is to be sent to the TSO userid.

#### Example

---

```
//DD1 DD TERM=TS
```

---

## Chapter 24. Output Formatting

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
<b>Output formatting</b>					
to any printer	COPIES FCB form-name on SYSOUT  UCS	COPIES FCB FORMS LINECT (JES2 only) UCS CONTROL	forms, copies, and linect on JOB JES2 accounting information	COPIES, FORMS, and LINECT on /*JOBPARM	
to 3800 Printing Sub- system, in addition to most of printer parameters	BURST  CHARS FLASH MODIFY DCB=OPTCD=J	BURST  CHARS FLASH MODIFY TRC		BURST on /*JOBPARM	
to 3211 Printer with indexing feature		INDEX (JES2 LINECT only)			
to punch	COPIES FCB form-name on SYSOUT DCB=FUNC=I	COPIES FCB FORMS			
of dumps on 3800 Printing Subsys- tem	CHARS=DUMP FCB=STD3	CHARS=DUMP FCB=STD3			

Figure 24-1. Output Formatting Task for Requesting Sysout Data Set Resources

# Output Formatting

## Output Formatting to Any Printer

To control the formatting of a printed sysout data set, code combinations of the following:

In any system:

```
//ddname DD SYSOUT=(class,form-name),COPIES=number,  
//      FCB=fcb-name,UCS=character-set-code  
  
//name   OUTPUT CONTROL=spacing,COPIES=number,FCB=fcb-name,  
//      FORMS=form-name,UCS=character-set-code
```

In a JES2 system:

```
//name   OUTPUT LINECT=number  
  
//jobname JOB    (,,,,,forms,copies,,linect)  
  
/*JOBPARM COPIES=number,FORMS=form-name,LINECT=number
```

Most of the formatting parameters can be coded on several statements. If coded more than once for a sysout data set, JES selects one parameter according to override rules and uses it.

Parameters coded on the JOB statement or /\*JOBPARM statement apply to all the sysout data sets in the job.

**3203 Printer Model 5 in a JES2 System:** JES2 treats the 3203 Model 5 the same as a 3211 Printer with the following exceptions:

- The universal character sets, specified in UCS parameters, for the 3203 Model 5 are the same as for the 1403 printer.
- The 3203 Model 5 does not support indexing; therefore, INDEX and LINDEX parameters are ignored.
- The installation cannot explicitly identify the 3203 Model 5 printer to JES2 during JES2 initialization. MVS passes the 3203 Model 5 identification to JES2 through the unit control block (UCB).

For further information on UCS and UCB, see *System-Data Administration*.

### Examples

---

```
//DD1   DD SYSOUT=(A,FMS3),COPIES=5,  
//      FCB=IMG7,UCS=AN  
  
//OTA   OUTPUT CONTROL=DOUBLE,COPIES=5,FCB=IMG7,  
//      FORMS=FMS3,UCS=AN
```

Use these parameters in any system.

---

```
//OTB   OUTPUT LINECT=60  
//J1    JOB    (,,,,,FMS3,5,,60)  
/*JOBPARM COPIES=5,FORMS=FMS3,LINECT=60
```

Use these parameters only in a JES2 system.

---

## Output Formatting to 3800 Printing Subsystem

To control the formatting of a sysout data set printed on a 3800 Printing Subsystem, code combinations of the following parameters and statements, in addition to the parameters used for printing on any printer.

In any system:

```
//ddname DD SYSOUT=class,BURST=value,CHARS=table-name,
//        COPIES=(,(group-value)),FLASH=overlay-name,
//        MODIFY=(module-name,trc),DCB=OPTCD=J

//name   OUTPUT BURST=value,CHARS=table-name,
//        COPIES=(,(group-value)),FLASH=overlay-name,
//        MODIFY=(module-name,trc),TRC=value
```

In a JES2 system:

```
/*JOBPARM BURST=value
```

Most of the formatting parameters can be coded on several statements. If coded more than once for a sysout data set, JES selects one parameter according to override rules and uses it.

The BURST parameter coded on the /\*JOBPARM statement applies to all sysout data sets printed on 3800 printers in the job.

### Copy Modification

For sysout data sets printed on a 3800, you can modify selected copies of output by specifying a copy modification module name in the MODIFY parameter. Copy modification allows printing predefined data on all pages of a copy or copies of the data set.

For example, you may want to vary column headings or explanatory remarks on different copies of the same printed page. Or, you may want to personalize copies with the recipient's name, address, and other information. Or, you may want to print blanks or certain characters, such as asterisks, to suppress the printing of variable data on particular copies of a page.

The predefined data is created as a copy modification module and stored in SYS1.IMAGELIB using the IEBIMAGE utility program. For information on using IEBIMAGE, see *Data Administration: Utilities*.

Copy modification is done with other printers by using short or spot carbons in the forms set.

### Character Arrangements

Specify in the CHARS parameter character-arrangement tables to be used when printing on a 3800.

For the names of tables for the 3800, see the *IBM 3800 Printing Subsystem Programmer's Guide*. The installation should maintain a list of the names of available tables.

**Modifying Character-Arrangement Tables:** Using the IEBIMAGE utility program, the installation can modify or construct character-arrangement tables and graphic character modification modules to substitute characters or use installation-designed characters.

## Output Formatting

**Dynamically Selecting Character-Arrangement Tables:** To select a character-arrangement table for each logical record in the sysout data set, the second character of each logical record must contain a trc character and you must code either of the following:

- TRC in the OUTPUT JCL statement
- OPTCD=J in the DD statement DCB parameter

For details on using the OPTCD subparameter, see the *IBM 3800 Printing Subsystem Programmer's Guide*.

**When Data Set Printed on 3800 or Other Printers:** You can code a UCS parameter even though a CHARS parameter is also coded; do this if the output might be printed on a 3800 or some other printer. If a printer other than the 3800 is used, the system uses the UCS parameter and ignores the CHARS parameter.

If UCS is coded and CHARS is not, and the sysout data set is printed on a 3800, the system uses the UCS value as the default value for the missing CHARS parameter.

### Examples

---

```
//DD8      DD  SYSOUT=B,BURST=YES,CHARS=(GS10,GU12),
//          COPIES=(, (5)),FLASH=BILL,MODIFY=(IMG9,1)

//OT4      OUTPUT BURST=YES,CHARS=(GS10,GU12),COPIES=(, (5)),
//          FLASH=BILL,MODIFY=(IMG9,1)
```

Use these parameters in any system.

---

```
/*JOBPARM BURST=Y
```

Use this statement only in a JES2 system.

---

## Output Formatting to 3211 Printer with Indexing Feature in a JES2 System

To request that output printed by JES2 on a 3211 Printer with the indexing feature be shifted from the normal page margins, code:

To indent left margin:  
//name OUTPUT INDEX=number

To move right margin:  
//name OUTPUT LINDEX=number

JES2 ignores these parameters if the output is printed on a device other than a 3211. To send a sysout data set to a 3211, specify the output class set aside by the installation for printing on a 3211.



## Examples

---

```
//OT10 OUTPUT INDEX=6  
//DD3 DD CLASS=W,OUTPUT=*.OT10
```

This example indents the left margin 5 spaces.

---

```
//OT11 OUTPUT LINDEX=9  
//DD3 DD CLASS=W,OUTPUT=*.OT11
```

This example moves the right margin in 8 spaces from the usual location.

---

## Output Formatting to Punch

To format punched output from sysout data sets, code:

```
//ddname DD SYSOUT=(class,form-name),COPIES=number,  
// FCB=fcb-name,DCB=FUNC=I  
  
//name OUTPUT COPIES=number,FCB=fcb-name,FORMS=form-name
```

### Interpretation of Punched Cards

Cards punched by a 3525 Card Punch are interpreted if JES processes the sysout data set and if the following is coded:

```
//ddname DD SYSOUT=class,DCB=FUNC=I
```

If the data set is punched on a different card punch, JES ignores the FUNC=I subparameter.

The installation can define a special output class for 3525 output.

Card interpretation by an external writer is an operator-specified function.

**Interpretation in a JES3 System:** Punched output may or may not be interpreted depending on the installation-defined standard for the output class.

## Examples

---

```
//DD17 DD SYSOUT=(Q,PUN6),COPIES=5,  
// FCB=IMG4,DCB=FUNC=I  
  
//OT3 OUTPUT COPIES=5,FCB=IMG4,FORMS=PUN6  
//DD18 DD SYSOUT=Q,OUTPUT=*.OT3,DCB=FUNC=I
```

---

## Output Formatting

### Output Formatting of Dumps on 3800 Printing Subsystem

You can request a high-density dump on the 3800 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

---

```
//SYSABEND DD SYSOUT=J,FCB=STD3,CHARS=DUMP  
  
//DUMPOT OUTPUT FCB=STD3,CHARS=DUMP  
//SYSABEND DD SYSOUT=J,OUTPUT=*.DUMPOT
```

---

## Chapter 25. Output Limiting

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Output limiting	OUTLIM		lines and cards on JOB JES2 accounting information	BYTES, CARDS, LINES, and PAGES on /*JOBPARM	BYTES, CARDS, LINES, and PAGES on /*MAIN

Figure 25-1. Output Limiting Task for Requesting Sysout Data Set Resources

### Output Limiting

To limit the number of logical records in a sysout data set, specify a maximum number of records to be written to a sysout data set or to all sysout data sets in a job by coding one of the following:

For one sysout data set:

```
//ddname DD SYSOUT=class,OUTLIM=number
```

For all sysout data sets in a job in a JES2 system:

```
//jobname JOB (,,lines,cards),...
/*JOBPARM BYTES=number
/*JOBPARM CARDS=number
/*JOBPARM LINES=number
/*JOBPARM PAGES=number
```

For all sysout data sets in a job in a JES3 system:

```
//*MAIN BYTES=number
//*MAIN CARDS=number
//*MAIN LINES=number
//*MAIN PAGES=number
```

To limit the size of an internal reader data set, use the DD OUTLIM parameter.

**Use of Limiting:** For example, a program enters an endless loop that includes a write instruction to a sysout data set. You can keep from printing a useless, huge listing by specifying a maximum number of records to be printed before the system abnormally terminates the step.

## Output Limiting

**Actions when Limit Exceeded in a JES3 System:** In a JES3 system, if the limit is exceeded, one of the following actions can be specified:

**WARNING:** JES3 issues a warning message to the operator.

**CANCEL:** JES3 terminates the job.

**DUMP:** JES3 terminates the job and dumps the step being executed when the limit was exceeded.

### Examples

---

```
//DD1 DD SYSOUT=T,OUTLIM=3000
```

Use this example in any system.

---

```
//JOBA JOB (,,4,2000),'T. KATZ'  
/*JOBPARM BYTES=40  
/*JOBPARM CARDS=2000  
/*JOBPARM LINES=4  
/*JOBPARM PAGE=400
```

Use these examples in a JES2 system.

---

```
//*MAIN BYTES=(40,WARNING)  
//*MAIN CARDS=(20,CANCEL)  
//*MAIN LINES=(4,DUMP)  
//*MAIN PAGES=(400,WARNING)
```

Use these examples in a JES3 system.

---

## **Part 6. Examples**

This part contains examples of sets of job control statements. Some are for useful processing, some show particular techniques. For examples of the job control statements needed to use utilities, see *Data Administration: Utilities* and for the statements needed to use service aids, see *Service Aids*.

### **Part 6 Contents**

<b>Chapter 26. Assemble, Linkedit, and Go</b>	<b>26-1</b>
<b>Chapter 27. Multiple Output</b>	<b>27-1</b>
<b>Chapter 28. Obtaining Output in a JES2 System</b>	<b>28-1</b>
<b>Chapter 29. Obtaining Output in a JES3 System</b>	<b>29-1</b>
<b>Chapter 30. Identifying Data Sets to the System</b>	<b>30-1</b>



## Chapter 26. Assemble, Linkedit, and Go

---

```

//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.
  - Link edit the object module, if the assembly did not result in a return code of 8 or higher: the step named LKED.
  - Execute the link edited module, if neither the assembly nor the linkage editing resulted in a return code of 8 or higher: the step named GO.
-

## Examples



## Chapter 27. Multiple Output

---

```

//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.

## Examples

**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.

### **Set 9**

The system-managed output data set is processed locally because of the MSGCLASS parameter on the JOB statement.

---

## Chapter 28. Obtaining Output in a JES2 System

---

```

/*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

```

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

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).
  - 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.

## Examples

- 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.
-

## Chapter 29. Obtaining Output in a JES3 System

---

```

//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

```

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

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 test printing on a 1403, 3211, or 3203-5 printer.
-

## Examples

## Chapter 30. Identifying Data Sets to the System

---

```
/*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.
-

## Examples



## Appendixes

### Appendixes Contents

<b>Appendix A. Indexed Sequential Data Sets</b>	<b>A-1</b>
Creating an Indexed Sequential Data Set	A-1
DSNAME Parameter	A-2
UNIT Parameter	A-2
VOLUME Parameter	A-2
LABEL Parameter	A-3
DCB Parameter	A-3
DISP Parameter	A-3
SPACE Parameter	A-4
Procedure when Allocation Error Occurs	A-4
Area Arrangement of an Indexed Sequential Data Set	A-4
Retrieving an Indexed Sequential Data Set	A-5
DSNAME Parameter	A-5
UNIT Parameter	A-6
VOLUME Parameter	A-6
DCB Parameter	A-6
DISP Parameter	A-6
Examples	A-7
<b>Appendix B. Generation Data Sets</b>	<b>B-1</b>
Relative Generation Numbers	B-1
Types of Data Sets in a GDG	B-1
Retrieval of GDG Data Sets	B-1
Building a GDG Base Entry	B-1
Data Set Label List	B-2
Creating a Generation Data Set	B-2
DSNAME Parameter	B-2
DISP Parameter	B-2
UNIT Parameter	B-3
VOLUME Parameter	B-3

## **Appendixes**

SPACE Parameter	B-3
LABEL Parameter	B-3
DCB Parameter	B-3
Retrieving a Generation Data Set	B-3
DSNAME Parameter	B-3
DISP Parameter	B-4
UNIT Parameter	B-4
VOLUME Parameter	B-4
LABEL Parameter	B-4
DCB Parameter	B-4
Deleting and Uncataloging Generation Data Sets	B-5
Submitting a Job for Restart	B-5
For Step Restart	B-5
For Checkpoint Restart	B-5
For Deferred Checkpoint Restart	B-5
Examples	B-5
<b>Appendix C. VSAM Data Sets</b>	<b>C-1</b>
Creating a VSAM Data Set	C-1
Retrieving Existing VSAM Data Set	C-1
DD Statement AMP Parameter	C-1

## Appendix A. 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:

- **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.
- **Prime area:** This area contains data and related track indexes. It exists for all indexed sequential data sets.
- **Overflow area:** This area contains overflow from the prime area when new data is added. It is optional.

Indexed sequential data sets must reside on direct access volumes. The data set can reside on more than one volume and the volumes may, in some cases, be on different types of devices.

### Creating an Indexed Sequential Data Set

One to three DD statements are used to define a new indexed sequential data set; each statement defines a different area.

#### Three DD statements

Define the areas in the following order:

1. Index area
2. Prime area
3. Overflow area

#### Two DD statements

Define the areas in the following order:

1. Index area
2. Prime area

Or

1. Prime area and, optionally, index area
2. Overflow area

## Appendix A. ISAM

### One DD statement

The statement defines 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:

```
DSNAME
UNIT
VOLUME
LABEL
DCB,
DISP
SPACE
```

**DSNAME Parameter:** The DSNAME parameter is required on any DD statement that defines a new temporary or permanent indexed sequential data set. Code:

```
//ddname DD DSNAME=name (INDEX)
//          DD DSNAME=name (PRIME)
//          DD DSNAME=name (OVLFLOW)
```

If you are using only one DD statement, code either:

```
//ddname DD DSNAME=name (PRIME)
//ddname DD DSNAME=name
```

When you reuse previously allocated space to create an indexed sequential data set, the DSNAME parameter must contain the name of the old data set to be overlaid.

**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. Do not code DEFER.

If the prime and index areas are defined on separate DD statements, request the same number of direct access devices for the prime area as volumes specified in the VOLUME parameter. Request only one direct access volume for an index area and one for an overflow area.

A DD statements for the index area or overflow area can request a device type different than the type requested on the other statements.

**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(s) in the VOLUME parameter or code VOLUME = REF = reference. In all cases, you can specify PRIVATE in the VOLUME parameter.

### Note:

- If a nonspecific volume request is used when creating a new indexed sequential data set and its DSNAME already exists on a volume eligible for allocation, the job will fail if the system places the new data set on that volume. However, if the old data set with the duplicate name is on a volume other than the one selected for the new data set, the new

data set is not affected and will be added to the volume. You can correct job failures caused by duplicate names by scratching the old data set or by renaming the new data set, then resubmitting the job.

- The system fails to allocate space for a new indexed sequential data set with a nonspecific volume request when none of the volumes eligible for allocation contain enough space.
- If the first volume selected by allocation to satisfy a request for a new indexed sequential data set does not contain enough space to satisfy the request, the system does not try to find another volume with enough space, if the request is one of the following:
  - For multiple volumes or units
  - The second, third, or subsequent DD statement defines the data set.

***LABEL Parameter:*** The LABEL parameter is needed only to specify a retention period, EXPDT or RETPD, or password protection, PASSWORD.

***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. On the other DD statements, refer to the DCB parameter on the first statement by coding:

DCB=\* . ddname

When reusing previously allocated space and recreating an indexed sequential 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	

***DISP Parameter:*** If you are creating a new data set and not reusing preallocated space, the DISP parameter is needed only if you want to:

Keep the data set	DISP=(,KEEP)
Catalog the data set	DISP=(,CATLG)
Pass the data set	DISP=(,PASS)

If you are reusing previously allocated space and recreating an indexed sequential data set, code DISP=OLD. The newly created data set will overlay the old one.

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 are on the same device type, use the access method services DEFINE command to catalog the data set. For details, refer to *Integrated Catalog Administration: Access Method Services Reference* or *VSAM Catalog Administration: Access Method Services Reference*.

## Appendix A. ISAM

**SPACE Parameter:** The SPACE parameter is required on any DD statement that defines a new indexed sequential data set. Either ask the system to assign the space or request specific tracks. If you use more than one DD statement to define the data set, each DD statement must request space in the same way.

**System Assignment of Space:** 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.

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.

**Specific Track Request:** The number of tracks requested must be equal to one or more whole cylinders. The address of the beginning track must be 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 be done only if the volume table of contents of the second and all succeeding volumes is contained in the first cylinder of each volume.

Use the index subparameter to indicate how many tracks the 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 state the size of the index, code the index subparameter.

### Procedure when Allocation Error Occurs

If a new indexed sequential data set is to reside on more than one volume and an error occurs during volume allocation, do the following 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 *Data Administration: Utilities*.

### Area Arrangement of an Indexed Sequential Data Set

When creating an indexed sequential data set, the arrangement of the areas is based on:

- The number of DD statements used to define the data set
- What area each DD statement defines

The system uses an additional criterion when the index area is not defined on a separate DD statement: Is an index size coded in the SPACE parameter on the DD statement that defines the prime area?

Figure A-1 on page A-5 illustrates the different arrangements that can result based on these criteria. In addition, it indicates what restrictions apply on the number and types of devices that can be requested.

Criteria			Restrictions on	Resulting
Number of DD statements	Area defined on DD statement	Index size coded?	Device Types and Number of Devices Requested	Arrangement of Areas
3	INDEX PRIME OVFLOW	-	None	Separate index, prime, and overflow areas.
2	INDEX PRIME	-	None	Separate index and prime areas. <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 for 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	The statement cannot request more than one device.	Prime area with embedded index area. <sup>2</sup>

<sup>1</sup> If both areas are on volumes on the same device type and if one of the cylinders allocated for the index area is only partially filled, the system establishes the overflow area in the unused portion of that cylinder.

<sup>2</sup> If the index area occupies at least one cylinder and if the unused portion of the index area is less than one cylinder, the unused portion is established as an overflow area. For a one-cylinder data set, no overflow area is established.

Figure A-1. Area Arrangement of ISAM Data Sets

### Retrieving an Indexed Sequential Data Set

If all areas of an existing indexed sequential data set are on volumes of the same device type, you can retrieve the entire data set with one DD statement. If the index or overflow is on a volume of a different device type, use two DD statements. If the index and overflow are 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. Index area
2. Prime area
3. Overflow area

The **only** DD statement parameters that you may code when retrieving an indexed sequential data set are:

DSNAME  
UNIT  
VOLUME  
DCB  
DISP

**DSNAME Parameter:** The DSNAME parameter is always required. Identify the data set by its name. Do not code INDEX, PRIME, or OVFLOW. If the data set was passed from a previous step, identify it by a backward reference.

## Appendix A. ISAM

**UNIT Parameter:** The UNIT parameter must be coded, unless the data set resides on one volume and was passed. Specify in the UNIT parameter the device type and the unit-count, if more than one device is required.

If the data set is on more than one volume but the volumes are for 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 areas are on different types of devices, code a DD statement for each different device type.

Another way to request a device is to code UNIT = AFF = ddname, where the referenced DD statement requests direct access.

**VOLUME Parameter:** The VOLUME parameter must be coded, unless the data set is 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 that they were coded on the DD statements used to create the data set.

**DCB Parameter:** The DCB parameter must always contain DSORG = IS or DSORG = ISU. Do not 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 is not completed in the processing program.

**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 indexed sequential data set, code DISP = OLD. If you specify DISP = SHR, the data set will not open correctly.

Optionally, you can specify a disposition in the second subparameter.

Area	Parameter	Comments
INDEX (coded only if index area is not on same device type as prime area)  First DD statement	DSNAME	Required. Code the same name as in the second DD statement.
	DISP	Required. Code the same value as in the second DD statement.
	UNIT	Required
	VOLUME	Required
	DCB	Required
PRIME; or PRIME with overflow; or PRIME with overflow and index  Second or only DD statement	DSNAME	Required
	DISP	Required. Specifies whether data set is being retrieved or updated.
	UNIT	Required, unless passed data set is being retrieved and all three areas are on one volume.
	VOLUME	Same requirement as UNIT. If coded, list volumes in the order in which they were defined.
	DCB	Required
OVFLOW (coded only if overflow area is not on same device type as prime area)  Third DD statement	DSNAME	Required. Code the same value as in the second DD statement.
	DISP	Required. Code the same value as in the second DD statement.
	UNIT	Required
	VOLUME	Required
	DCB	Required

Figure A-2. DD Parameters for Retrieving or Extending an ISAM Data Set



## Examples

---

```
//ISAMJOB JOB      , ,MSGLEVEL=(1,1),PERFORM=25
//STEP1  EXEC     PGM=INCLUDE
//DD1    DD       DSNNAME=DATASET1(INDEX),DISP=(NEW,KEEP),UNIT=3330,
//          VOLUME=SER=777777,SPACE=(CYL,(10),,CONTIG),
//          DCB=(DSORG=IS,RECFM=F,LRECL=80,RKP=1,KEYLEN=8)
//          DD     DSNNAME=DATASET1(PRIME),DISP=(NEW,KEEP),UNIT=3330,
//          VOLUME=REF=*.DD1,SPACE=(CYL,(25),,CONTIG),DCB=*.DD1
//          DD     DSNNAME=DATASET1(OVFLOW),DISP=(NEW,KEEP),UNIT=3330,
//          VOLUME=REF=*.DD1,SPACE=(CYL,(25),,CONTIG),DCB=*.DD1
```

This example creates an indexed sequential data set on one 3330 volume.

---

```
//RETRISAM JOB      , ,MSGLEVEL=(1,1),PERFORM=25
//STEP1  EXEC     PGM=RETRIEVE
//DDISAM DD       DSNNAME=DATASET1,DCB=DSORG=IS,UNIT=3330,DISP=OLD,
//          VOLUME=SER=777777
```

This example job shows the DD statements needed to retrieve the indexed sequential data set created in the first example.

---

```
//ISAMJOB JOB      , ,MSGLEVEL=(1,1),PERFORM=25
//STEP1  EXEC     PGM=IEFISAM
//DDISAM DD       DSNNAME=DATASET2(INDEX),DISP=(NEW,KEEP),UNIT=3330,
//          VOLUME=SER=888888,SPACE=(CYL,10,,CONTIG),DCB=(DSORG=IS,
//          RECFM=F,LRECL=80,RKP=1,KEYLEN=8)
//          DD     DSNNAME=DATASET2(PRIME),DISP=(,KEEP),UNIT=3350,
//          VOLUME=SER=999999,SPACE=(CYL,10,,CONTIG),DCB=*.DDISAM
//          DD     DSNNAME=DATASET2(OVFLOW),DISP=(,KEEP),UNIT=3350,
//          VOLUME=SER=AAAAAA,SPACE=(CYL,10,,CONTIG),DCB=*.DDISAM
```

This job creates an indexed sequential data set on one 3330 and two 3350 volumes.

---

```
//RERISAM JOB      , ,MSGLEVEL=(1,1),PERFORM=25
//STEP1  EXEC     PGM=IEFISAM
//DDISAM DD       DSNNAME=DATASET2,DCB=DSORG=IS,DISP=OLD,UNIT=3330,
//          VOLUME=SER=888888
//          DD     DSNNAME=DATASET2,DCB=DSORG=IS,DISP=OLD,UNIT=(3350,2),
//          VOLUME=SER=(999999,AAAAAA)
```

This job shows the DD statements needed to retrieve the indexed sequential data set created in the previous example.

---



## Appendix B. 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 DD statement DSNAMES parameter and follow the group name with a relative generation number.

**Relative Generation Numbers:** When creating 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 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.

**Types of Data Sets in a GDG:** 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 in a GDG can have like or unlike DCB attributes and data set organizations.

**Retrieval of GDG Data Sets:** 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 GDG Base Entry:** Before defining the first generation data set, build a generation data group base entry in a VSAM, OS CVOL, or ICF catalog. This must provide for as many generation data sets, up to 255, as you would like to have in the GDG. 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 *Integrated Catalog Administration: Access Method Services Reference* or *VSAM Catalog Administration: Access Method Services Reference*.

## Appendix B. GDG

**Data Set Label List:** Another requirement for a GDG is a data set label list. The system uses this label to refer to DCB attributes when you define a new generation data set. The two ways to satisfy this requirement are:

- Create a model data set label on the same volume as the catalog before defining the first generation data set.
- 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:** A model data set label can be used by every GDG whose indexes are on the same volume. To create the model data set label, code a DD statement with the following parameters:

- Specify the same volume as the GDG base. This ensures that the system can always refer to a data set label on the same volume as the catalog.
- Identify the data set with the same or a different name as the name for the GDG. 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.
- Code any of these DCB attributes: DSORG, OPTCD, BLKSIZE, LRECL, KEYLEN, and RKP.

When creating a generation data set, specify the name of the model in the DCB parameter.

**Referring the System to a Cataloged Data Set:** If a cataloged data set is on the same volume as the GDG index and that data set will exist as long as you are adding data sets to the GDG, you need not create a model data set label. When creating a generation data set, specify the name of the cataloged data set in the DCB parameter.

### Creating a Generation Data Set

When defining a new generation data set, always code the DSNAME, DISP, and UNIT parameters. Optional parameters are VOLUME, SPACE, LABEL, and DCB.

**DSNAME Parameter:** In the DSNAME parameter, code the name of the GDG followed by a number, +1 to +255, in parentheses. If this is the first data set being added to a GDG in the job, code +1 in parentheses. Each time in the job you add a data set to the same GDG, increase the number by one.

When referring to this data set later in the job, use the relative generation number used in creating it. 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 a relative generation number makes the actual generation number exceed G9999.

**DISP Parameter:** Assign new generation data sets a status of new and a disposition of catalog: DISP=(NEW,CATLG).

**UNIT Parameter:** The UNIT parameter is required for a new generation data set unless VOLUME=REF=reference is coded. In the UNIT parameter, identify the type of device wanted.

**VOLUME Parameter:** Assign a volume in the VOLUME parameter, or omit the VOLUME parameter and let the system assign the volume. The VOLUME parameter can request a private volume, PRIVATE, and more than one volume in the volume count.

**SPACE Parameter:** Code the SPACE parameter when the generation data set is to reside on a direct access volume.

**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 is to reside on a tape volume and is not the first data set on the volume, specify a data set sequence number.

**DCB Parameter:** If a model data set label has the same name as the GDG and if the label contains all the attributes for this generation data set, omit 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 GDG and if the label contains all the attributes for this generation data set, code DCB=dsname. If some attributes are missing from the label or if you want to override some attributes, code DCB=(dsname,list of attributes).

If a model data set label does not exist, you must use the label for a cataloged data set on the same volume as the GDG index. Code DCB=dsname. If some attributes are missing from the label, or if you want to override some attributes, code DCB=(dsname,list of attributes).

### Retrieving a Generation Data Set

To retrieve a generation data set, always code the DSNNAME and DISP parameters. Optional parameters are the UNIT, LABEL, and DCB.

**DSNNAME Parameter:** Use the DSNNAME parameter to retrieve a single generation data set or all of the generation data sets in the GDG.

**Retrieving a Single Generation Data Set:** To retrieve a single generation data set, code in the DSNNAME parameter the name of the generation data group followed by a number, 0 to 9999, in parentheses. The number indicates which generation data set is to be retrieved. To retrieve the most recent data set, code a zero. If the first character is zero, the remaining characters must be zero or blanks.

To retrieve data sets created before the most recent data set, code a minus value, -1 to -255. The value of nnn indicates the relation of the desired data set to the most current data set: (-1) refers to the data set created immediately before the most recent data set; (-2) refers to the data set created before the data set identified by (-1). For example:

PAYROLL	Name of the GDG
DSNNAME=PAYROLL(0)	This week's generation data set
DSNNAME=PAYROLL(-1)	Last week's generation data set
DSNNAME=PAYROLL(-2)	Generation data set of two weeks ago

## Appendix B. GDG

Relative generation numbers are maintained by the system only when generation data sets are specified using relative generation numbers.

*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 produces unpredictable results.

*Retrieving All Generation Data Sets:* To retrieve all generations of a GDG as a single data set, specify the GDG name without a generation number in the DSNNAME parameter; this is called a GDG ALL request. For example:

**DSNNAME=PAYROLL** For all generations

To use a GDG ALL request, the DCB attributes and data set organization of all generations must be identical.

The system treats a GDG ALL request as a concatenation of all existing data sets in the GDG, starting with the most recent data set and ending with the oldest. All older generations have unit affinity to the newest data set.

For a GDG on tape, when you use a GDG ALL request and specify parallel mounting in the UNIT parameter, the system mounts all volumes of only the **first** generation.

For a GDG on direct access, including MSS, when you use a GDG ALL request and specify parallel mounting in the UNIT parameter, the system mounts all volumes of **all** generations.

*DISP Parameter:* Always code the DISP parameter. The first subparameter of the DISP parameter must be OLD, SHR, or MOD. If you code MOD for a generation data set and the specified relative generation does not exist in the catalog, the system changes the status to NEW.

A normal termination disposition is optional when retrieving a generation data set but is required in a GDG ALL request. Do not code PASS in a GDG ALL request.

*UNIT Parameter:* Code the unit-count subparameter in the UNIT parameter when you want more than one device assigned to the data set. Or, if the data set resides on more than one volume and you want as many devices as there are volumes, code P in the UNIT parameter.

*VOLUME Parameter:* Use the VOLUME parameter to request a private volume, PRIVATE, and to indicate that more volumes might be required, volume count. Do not specify a volume serial number for an old generation data set.

*LABEL Parameter:* Code the LABEL parameter when the data set is 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 is obtained from the catalog.

*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.

## Deleting and Uncataloging Generation Data Sets

In a multiple-step job, catalog or uncatalog generation data sets using the DD DISP parameter. Do not use the IEHPROGM utility program or a user program. Because system 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.

If a DD statement in a multiple-step job tries to delete or uncatalog any generation data set **except** the oldest in a GDG, 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 older generations in later job steps.

## Submitting a Job for Restart

Certain rules apply when you refer to generation data sets in a job submitted for restart using the RESTART parameter on the JOB statement.

**For Step Restart:** To refer to generation data sets that were created and cataloged in steps before the restart step, use their present relative generation numbers. 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 when created 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), you can, during checkpoint restart, refer to these data sets and generation data sets created and cataloged in steps before the restart step by 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 GDG. 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 numbers.

## Examples

---

```
//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)
```

This step shows DD statements used to add three data sets to a GDG.

DD1 and DD2 do not include the DCB parameter because a model data set label exists on the same volume as the GDG index and has the same name as the GDG: 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.

---

## Appendix B. GDG

---

```
//JWC   JOB   , 'J. GRIFFIN-KEENE'  
//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
```

This job shows the DD statements needed to retrieve the generation data sets defined in the first example, when the GDG contains no other generation data sets.

---



### Appendix C. VSAM Data Sets

Virtual storage access method (VSAM) can be used for data sets on direct access storage. Because VSAM is different from the other access methods, certain DD parameters and subparameters are different for VSAM data sets.

**Creating a VSAM Data Set:** Use access method services commands to create a VSAM data set. You cannot use a DD statement.

**Retrieving Existing VSAM Data Set:** To request a cataloged VSAM data set, code a DD statement in the form:

```
//ddname DD DSNAME=dsname,DISP=OLD  
//ddname DD DSNAME=dsname,DISP=SHR
```

**Note:** VSAM data sets cannot be passed within a job.

DD statement parameters that can be used without modification are explained in Figure C-1 on page C-2. Parameters that should not be used or should be used only with caution are explained in Figure C-2 on page C-3.

**DD Statement AMP Parameter:** VSAM has one DD statement parameter of its own: AMP. The AMP parameter takes effect when the data set defined by the DD statement is opened.

## Appendix C. VSAM

Parameter	Subparameter	Comment
<b>AMP</b>		This parameter has subparameters for: <ol style="list-style-type: none"> <li>1. Overriding operands specified with the ACB, EXLST, or the GENCB macro instructions</li> <li>2. Supplying operands missing from the ACB or GENCB macro instruction</li> <li>3. Indicating checkpoint/restart options</li> <li>4. Indicating options when using ISAM macro instructions to process a key-sequenced data set</li> <li>5. Indicating that the data set is a VSAM data set when the DD statement specifies unit and volume information or DUMMY</li> <li>6. Indicating that VSAM is to supply storage dumps of the ACBs that identify the DD statement</li> </ol>
<b>DDNAME</b>	<i>ddname</i>	No special considerations for VSAM.
<b>DISP</b>	<b>SHR</b>  <b>OLD</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 Administration Guide</i> for a full description of data-set sharing.  No special considerations for VSAM.
<b>DSNAME</b>	<i>dsname</i>	Names the VSAM cluster to which the data set belongs.
<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>device number</i>  <i>type</i>  <i>group</i>  <i>p</i>  <i>unit count</i>  <b>DEFER</b>	Must be the device number of a valid device for VSAM (2305, 3330V, 3330, 3340, 3344, 3350, 3375, or 3380). If not, OPEN will fail  Must be a type supported by VSAM (2305, 3330, 3330V, 3340, 3350, 3375, or 3380). If not, OPEN will fail.  Must be a group supported by VSAM. If not, OPEN will fail.  System must have 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.  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.  No special considerations for VSAM.
<b>VOLUME</b>	<b>PRIVATE</b> <b>SER</b>	No special considerations for VSAM.  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 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 needed, if at least one of the units allocated to the data set cannot be shared or the unit count is equal to the total number of volumes allocated to the data set. A unit cannot be shared when the 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 C-1. DD Parameters to Use when Processing VSAM Data Sets

Parameter	Subparameter	Comment
<b>BURST</b>		Not applicable.
<b>CHARS</b>		Not applicable.
<b>CHKPT</b>		VSAM ignores CHKPT
<b>COPIES</b>		Not applicable.
<b>DATA</b>		Not applicable.
<b>DCB</b>	All	Not applicable.
<b>DEST</b>		Specify DEST only with the SYSOUT parameter.
<b>DISP</b>	<b>CATLG</b>	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.
	<b>DELETE</b>	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.
	<b>MOD</b>	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.
	<b>KEEP</b>	Because KEEP is implied for VSAM data sets, it need not be coded.
	<b>NEW</b>	VSAM data spaces are initially allocated as a result of the access method services DEFINE command. If NEW is specified, the system allocates space, but 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.
	<b>UNCATLG</b>	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.
	<b>PASS</b>	Not applicable. 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.
<b>DLM</b>		Not applicable.
<b>DSNAME</b>	<i>dsname(area-name)</i> <i>dsname(generation)</i> <i>dsname(member)</i>	VSAM uses the dsname. An area-name, generation number, or member name is ignored, if coded with the dsname.
	All temporary <i>dsnames</i>	No not code a temporary dsname for a VSAM data set.
	All backward	Do code code backward references to VSAM data sets. If the object referred to is a cluster and
	DD references of the form <i>*.ddname</i>	data set and index reside on unlike devices, the results of a backward DD reference are unpredictable.

Figure C-2 (Part 1 of 2). DD Parameters to Avoid when Processing VSAM Data Sets

## Appendix C. VSAM

Parameter	Subparameter	Comment
<b>FCB</b>		Not applicable.
<b>FLASH</b>		Not applicable.
<b>LABEL</b>	<b>BLP, NL, NSL</b> <b>IN</b> <b>OUT</b> <b>NOPWREAD</b> <b>PASSWORD</b> <b>SL, SUL</b>	Not applicable. Not applicable. Not applicable. The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter. The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter. 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>		Not applicable.
<b>MSVGP</b>		Not applicable.
<b>SPACE</b>		VSAM data spaces are initially allocated as a result of the access method services DEFINE command. If SPACE is specified, the system allocates space, but it 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	Not applicable.
<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>  <i>vol-seq-number</i> <i>volume-count</i>	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. Results are unpredictable. Not applicable because this subparameter gives the number of nonspecific volumes. All VSAM volumes must be specifically defined.
*	Not applicable.	

Figure C-2 (Part 2 of 2). DD Parameters to Avoid when Processing VSAM Data Sets

## Index

- \*
    - \*
      - use in identifying in-stream data set 12-5
  - /
  - /\*DEL statement
    - use of 23-4
  - /\*EOF statement
    - use of 23-4
  - /\*MESSAGE statement
    - use of 7-2
  - /\*NOTIFY statement
    - use of 7-5
  - /\*PRIORITY statement
    - use of 11-3
  - /\*PURGE statement
    - use of 23-4
  - /\*ROUTE statement
    - use of 23-1
  - /\*SCAN statement
    - use of 23-4
  - /\*SETUP statement
    - use of 15-27
  - /\*SIGNOFF statement
    - use of 6-4
  - /\*SIGNON statement
    - use of 6-4
  - /\*DATASET statement
    - use of 12-6
  - /\*ENDDATASET statement
    - use of 12-6
  - /\*ENDPROCESS statement
    - use of 10-14
  - /\*OPERATOR statement
    - use of 7-2
  - /\*PROCESS statement
    - use of 10-14
    - when requesting processor 9-10
- A
- abnormal termination
    - See termination, abnormal
  - ACB (access method control block)
    - values for data set processing 13-6
  - access method control block
    - See ACB (access method control block)
  - ACCODE parameter
    - use of 14-2
  - accounting information
    - to identify account 4-3
  - ACMAIN parameter
    - use of 7-5
  - affinity
    - chart of unit and volume affinities 15-7
    - for multivolume data sets 15-8
    - interaction of unit and volume affinities 15-6
    - unit 15-6
      - explicit 15-6
      - implied 15-6
      - to subsystem data set 16-5
      - when requesting extended data sets 15-8
    - volume 15-6, 15-18
      - explicit 15-18
      - implicit 15-18
  - aging
    - See priority, aging
  - allocation, task for requesting data set resources
    - chart of 15-1
    - described 15-1-15-28
    - dynamic 15-28
      - example 15-28
    - of device 15-1
      - affected by device status 15-2
      - concurrent 15-3
      - examples 15-9
      - in JES3 system 15-11
      - number allocated 15-4
      - requesting more than one 15-3
    - of direct access space 15-21
      - example 15-23, 15-24
    - of tape or direct access volume 15-15
      - examples 15-9
      - for indexed sequential data set, if error occurs A-4
    - of virtual I/O 15-24
      - examples 15-25, 15-26
      - with deferred volume mounting 15-27
      - example 15-27
      - with volume premounting 15-27
        - example 15-28
  - AMP parameter
    - use of 13-6, C-1
  - APG (automatic priority group)
    - default dispatching priority 11-4
  - automatic priority group
    - See APG (automatic priority group)
- B
- base control program
    - See BCP (base control program)
  - BCP (base control program)
  - binary synchronous communication
    - See BSC (binary synchronous communication)
  - BSC (binary synchronous communication)
    - use of 6-3, 6-4

# Index

- BURST parameter
  - use of 24-3
- BYTES parameter
  - use of 7-4, 10-10, 10-14, 25-1
- C
- CANCEL subparameter
  - use of 25-2
- CARDS parameter
  - use of 7-4, 10-10, 10-14, 25-1
- cards subparameter
  - use of 25-1
- cataloged procedures
  - See procedures, cataloged and in-stream
- cataloging, data set
  - not performed as requested 17-4
  - requesting 17-4
  - use of 17-4
  - when cataloged data set updated 17-4
- catalogs
  - in JES3 allocation 15-12
  - of job control procedures (SYS1.PROCLIB)
    - private, of data sets
      - use of 12-7
    - system, of data sets
      - use of 12-7
    - volume for
      - allocation and deallocation of 12-7
- character-arrangement tables
  - modifying 24-3
  - selecting dynamically 24-4
  - specifying 24-3
- CHARS parameter
  - use of 24-3, 24-6
- checkpointing
  - of job execution 5-2
  - of multivolume data sets 16-4
  - of sysout data set 21-10
- CHKPT macro
  - in restarts 5-3
- CHKPT parameter
  - use of 16-4
- CHNSIZE parameter
  - use of 21-10, 23-1
- CKPTLINE parameter
  - use of 21-10
- CKPTLNS parameter
  - use of 21-10
- CKPTPAGE parameter
  - use of 21-10
- CKPTPGS parameter
  - use of 21-10
- CKPTSEC parameter
  - use of 21-10
- class
  - job
    - described 11-2
    - for copying job 6-3
    - in holding job 6-1
    - use of to control performance 11-2
- output
  - assigning data set to 18-2
  - printing of 21-5
  - use of 21-5
- CLASS parameter
  - for copying job 6-3
  - in holding job 6-1
  - to suppress sysout output 21-10
  - use of 7-7, 10-12, 11-2, 18-2, 21-8
  - when requesting processor 9-10
- CNTL parameter
  - use of 16-5
- CNTL statement
  - use of 16-5
- code
  - See return code
- command statement
  - use of 7-2
- comments
  - use of 7-2
- communication, task for entering jobs
  - chart of 7-1
  - described 7-1-7-8
  - from functional subsystem to programmer 7-6
    - example 7-6
  - from JCL to operator 7-2
    - examples 7-2
  - from JCL to program 7-3
    - examples 7-3
  - from JCL to programmer 7-2
    - examples 7-3
  - from JCL to system 7-2
    - examples 7-2
  - from system to operator 7-3
    - examples 7-4
  - from system to TSO userid 7-4
    - examples 7-5
  - from TSO userid to system 7-5
    - examples 7-5
  - through job log 7-6
    - examples 7-7, 7-8
- COMPACT parameter
  - use of 23-1
- COND parameter
  - examples 10-5
  - relationship of JOB and EXEC statement COND
    - parameters 10-3
  - use of 10-2
  - use of to force step execution 10-3
- CONTROL parameter
  - use of 24-2
- controlling
  - data sets 13-1
- COPIES parameter
  - use of 24-2, 24-3, 24-5
- copies subparameter
  - use of 24-2
- copying

- input stream 6-3
  - of data set name 12-5
- D**
- data control block
    - See DCB (data control block)
  - DATA parameter
    - use in identifying in-stream data set 12-5
  - data sets
    - cataloged
      - deletion of 17-3
      - generation data sets 17-4
      - on MSS 15-20
      - placing in catalog 17-4
      - removing from catalog 17-4
      - specifying CATLG disposition for 17-4
      - unit and volume information for 15-5
      - volume references to 15-5
    - dummy
      - effect of 16-1
      - nullifying 16-2
      - to suppress sysout output 21-9
      - use of 16-2
    - exclusive use of 13-2
    - held
      - releasing 21-8
      - requesting 21-8
    - multivolume 15-18
      - checkpointing 16-4
      - considerations when allocating 15-18
    - passed
      - demounting of volume 17-11
      - effect on volume retention 17-11
      - unit and volume information for 15-6
    - permanent 12-2
    - postponed definition 16-2
      - concatenating 16-3
      - references to 16-3
      - use of 16-3
    - securing control of 13-1
    - shared use of 13-2
    - sysout
      - grouping 21-6
      - size of 21-5
    - system-managed
      - described 7-6
    - temporary 12-3
      - deletion of 17-3
      - use of VIO for 15-24
  - data-set-sequence-number
    - use of 12-10
  - DCB (data control block)
    - values for data set processing 13-5
    - values for sysout data set processing 19-1
    - values from cataloged data sets 13-5
    - values from earlier DD statements 13-5
  - DCB parameter
    - use of 13-5, 19-1, 24-3, 24-4, 24-5, A-3, A-6, B-3, B-4
  - DDNAME parameter
    - use of 16-2
  - deadline
    - execution by 5-4
  - DEADLINE parameter
    - in deadline scheduling 5-4
    - in periodic scheduling 5-4
  - deallocation
    - of data set, volume, and device 17-1
  - DEFER subparameter
    - use of 15-27
  - deferred volume mounting
    - specifying 15-27
  - deleting, data set
    - if data set cataloged 17-3
    - if expiration date unexpired 17-3
    - not performed when data set uncataloged 17-4
    - of generation data set B-5
    - of temporary data set 17-3
    - requesting 17-3
  - delimiter statement
    - use of 12-5
  - dependencies, before step execution
    - external 5-5
  - dependent job control
    - See DJC (dependent job control)
  - dependent job net
    - testing of 5-5
    - use of 5-5
  - description, task for requesting data set resources
    - chart of 13-1
    - described 13-1-13-6
    - of data attributes 13-5
      - examples 13-5, 13-6
    - of status 13-1
      - examples 13-2
  - description, task for requesting sysout resources
    - chart of 19-1
    - described 19-1
    - of data attributes 19-1
      - example 19-1
  - DEST parameter
    - use of 23-1
  - destination control, task for requesting sysout resources
    - chart of 23-1
    - described 23-1-23-6
    - to another processor 23-4
      - example 23-4
    - to internal reader 23-4
      - example 23-5
    - to local or remote device or to another node 23-1
      - examples 23-3
      - in JES2 network 23-2
      - in JES3 network 23-3
    - to terminal 23-6
      - example 23-6
  - destinations
    - default 23-2

# Index

- multiple 23-2
- device
  - allocation of 15-1
  - management in JES3 system 15-11
  - number allocated 15-4
  - specifying as destination for sysout data set 23-1
- directory
  - of PDS 9-1
- DISP parameter
  - use of 13-1, 17-2, A-3, A-6, B-2, B-4, B-5
  - use when data set is cataloged 12-7
- DJC (dependent job control)
  - use of 5-5
- DLM parameter
  - use of 12-5
- documenting
  - job and its resource requirements 7-2
- DPRTY parameter
  - use of 11-4
- DSID parameter
  - use of 12-6, 18-2
- DSNAME parameter
  - use of 12-2, 15-15, A-2, A-5, B-2, B-3
- DUMMY parameter
  - use of 16-1, 21-9
  - with SUBSYS parameter 16-5
- DUMP subparameter
  - use of 25-2
- dumping
  - after error 10-14
  - formatting of 24-6
  - high-density 10-15
- dynamic
  - allocation 15-28
  - deallocation 17-1
- DYNAMNBR parameter
  - use of 15-28

## E

- end processing, task for requesting data set resources
  - chart of 17-1
  - deallocation 17-1
    - example 17-2
  - described 17-1-17-11
  - disposition of data set 17-2
    - bypassing 17-6
    - cataloging 17-4
    - default 17-6
    - deletion 17-3
    - effect of device type 17-3
    - examples 17-7, 17-9
    - keeping 17-3
    - passing 17-5
    - uncataloging 17-4
    - when no abnormal termination disposition coded 17-2
  - disposition of volume 17-10
    - examples 17-10, 17-11

- of removable volumes 17-10
- release of unused direct access space 17-9
  - example 17-9
- end processing, task for requesting sysout resources
  - chart of 22-1
  - deallocation 22-1
    - example 22-1
    - described 22-1
- entering jobs
  - task in job control 2-1
  - tasks
    - chart of 3-2
- errors
  - scanning JCL for 10-12
- EVEN subparameter
  - examples 10-10
  - use of to force step execution 10-3
- event, external
  - holding job for 6-1
- examples
  - of assemble, linkedit, and go 26-1
  - of identifying data sets to system 30-1
  - of multiple output 27-1
  - of obtaining output in JES2 system 28-1
  - of obtaining output in JES3 system 29-1
- execution, task for entering jobs
  - at remote node 5-7
    - considerations for 5-8
    - examples 5-8
  - chart of 5-1
  - deadline or periodic 5-4
    - examples 5-4
    - use of 5-4
    - described 5-1-5-8
  - of procedure 5-2
    - examples 5-2
  - of program 5-1
    - examples 5-2
  - required task 3-1
  - when dependent on other jobs 5-5
    - examples 5-6
  - when restarting and with checkpointing 5-2
    - examples 5-3
- EXPDT subparameter
  - use of 17-8
- expiration date, for data set
  - deleting before 17-8
  - effect on disposition 17-8
  - requesting 17-8
  - when unexpired 17-3
- extents
  - in allocation of direct access space 15-22
- external writer
  - See writer, external

## F

- FAILURE parameter
  - in restarts 5-3



FCB parameter  
 use of 24-2, 24-5, 24-6

FETCH parameter  
 use of 7-3

FLASH parameter  
 use of 24-3

FORMDEF parameter  
 use of 21-11

FORMS parameter  
 use of 24-5

forms subparameter  
 use of 24-2

FREE parameter  
 use of 17-1, 22-1

## G

GDG (generation data group)  
 building base entry for B-1  
 cataloging data sets of 17-4  
 creating model data set label B-2  
 data set label list B-2  
 identifying 12-3  
 referring to cataloged data set for label B-2  
 types of data sets in B-1

generation data group  
 See GDG (generation data group)

generation data sets  
 creating B-2  
 deleting and uncataloging B-5  
 described B-1-B-6  
 examples B-5  
 retrieving B-1, B-3  
 rules when submitting job for restart B-5

graphic character modification modules  
 modifying 24-3

GROUP parameter  
 use of 8-1

GROUPID parameter  
 use of 21-6

grouping, sysout data sets  
 demand setup 21-6  
 requesting 21-6  
 subgroups 21-6

## H

hard-copy log  
 described 7-6

high-density dumps  
 requesting 24-6

HOLD parameter  
 in holding job 6-1  
 use of 21-8

holding  
 job entrance 6-1  
 of sysout data set 21-8  
 releasing 21-8

use of 21-8

## I

I/O-to-processing ratio  
 use of 11-5

identification, task for entering jobs  
 chart of 4-1  
 described 4-1-4-4  
 of account 4-3  
 examples, for local execution 4-3  
 examples, for remote execution 4-4  
 for local execution 4-3  
 for remote execution 4-4

of job 4-2  
 examples 4-2  
 identified 2-1  
 required task 3-1

of procedure 4-2  
 examples 4-3

of programmer 4-4  
 examples 4-4

of step 4-2  
 examples 4-2  
 identified 2-1

identification, task for requesting data set resources  
 as TCAM message data set 12-10  
 example 12-10  
 by location on tape 12-10  
 examples 12-10  
 chart of 12-1  
 described 12-1-12-11  
 from or to terminal 12-11  
 example 12-11

of data set 12-2  
 examples for generation data set 12-3  
 examples for indexed sequential data set 12-3,  
 12-5  
 examples for partitioned data set 12-2, 12-4  
 examples for permanent data set 12-2  
 examples for temporary data set 12-4  
 examples when copying data set name 12-5

of data set on 3450 Diskette Input/Output  
 Unit 12-6  
 example 12-7

of in-stream data set 12-5  
 examples 12-6

through catalog 12-7  
 examples 12-8

through label 12-8  
 examples 12-9

identification, task for requesting sysout resources  
 as a sysout data set 18-1  
 examples 18-1  
 chart of 18-1  
 described 18-1-18-3  
 of data set on 3450 Diskette Input/Output  
 Unit 18-2  
 example 18-3

# Index

- of output class 18-2
  - examples 18-2
- IEBIMAGE utility program
  - use for character-arrangement tables 24-3
  - use in updating SYS1.IMAGELIB 24-3
- IEBUPDTE utility program
  - use of 9-5
- IEFBR14 program
  - considerations when using 10-13
  - described 10-13
  - use in testing 10-13
- IEHPROGM utility program
  - use of B-5
- IN subparameter
  - use of 14-3, 14-4
- in-stream
  - See input stream
- in-stream procedures
  - See procedures, cataloged and in-stream
- independent mode, processor
  - requesting 9-9
- index area
  - defined A-1
  - identifying 12-3, 12-4
- INDEX parameter
  - use of 12-3, 24-4
- indexed sequential data set
  - area arrangement for A-4
  - creating A-1
  - described A-1-A-7
  - examples A-7
  - identifying 12-3, 12-4
  - parameters for retrieving or extending A-6
  - retrieving A-5
  - specific track requests for 15-24
  - system assigned space requests for 15-23
  - when allocation error occurs A-4
- indexing
  - of sysout data set margins 24-4
- input control, task for entering jobs
  - by copying input stream 6-3
    - examples 6-3
  - by holding job entrance 6-1
    - examples 6-2
    - use of 6-2
  - by holding local input reader 6-2
    - example 6-2
  - chart of 6-1
  - described 6-1-6-4
  - from remote work station 6-3
- input stream
  - defined 2-2
  - devices for 2-2
  - example 2-2
  - identifying data sets in 12-5
- integrity processing
  - chart of 13-4
  - defined 13-1
  - for other than permanent data sets 13-3
  - for permanent data sets 13-3
- of data sets 13-2
- interpreting, punched sysout data set
  - requesting 24-5
- INTRDR subparameter
  - use of 23-4
- invalid
  - syntax, scanning for 10-12
- IORATE parameter
  - use of 11-5
- J
- JCL statements
  - in job control iii
  - purposes of 1-1
- JCLTEST subparameter
  - use of 10-12
- JESDS parameter
  - use of 7-6, 7-7
- JES2
  - in job control iii
  - statements
    - in jobs 2-4
    - purposes of 1-2
- JES3
  - in job control iii
  - statements
    - in jobs 2-4
    - purposes of 1-2
- job control language statements
  - See JCL statements
- job input control
  - See input control, task for entering jobs
- job log
  - described 7-6
  - execution time messages in log 10-11
  - for communication from JCL to programmer 7-2
  - output class for 7-6
  - printing of 7-6
  - printing with sysout data sets 7-7
- JOBCAT catalog
  - use of 12-7
- JOBLIB
  - See libraries, private (JOBLIB or STEPLIB)
- jobname
  - to identify job 4-2
- jobs
  - background
    - defined 7-4
  - batch
    - defined 7-4
  - control of 2-1-2-4
  - defined 2-1
  - entering 2-1
  - examples 2-1
  - examples with JES2 or JES3 statements 2-4
  - predecessor 5-5
  - processing 2-4
  - requesting resources 2-4

- steps of 2-1
- successor 5-5
- JOURNAL parameter
  - in restarts 5-3
- JSTTEST subparameter
  - use of 10-12

**K**

- keep disposition
  - for tape volume 17-10
- keeping, data set
  - requesting 17-3
  - when data set uncataloged 17-4

**L**

- LABEL parameter
  - use of 12-8, A-3, B-3, B-4
- label, data set
  - for cataloged or passed data set 12-9
  - for nonspecific volume request 12-9
  - for specific volume request 12-9
  - use of 12-8
- libraries
  - concatenating 9-4
  - defined 9-1
  - private (JOBLIB or STEPLIB) 5-1, 9-1
    - adding to 9-2
    - creating 9-2
    - retrieving 9-3
    - use of 9-2
  - procedure 9-5
  - updating 9-5
  - residence for executable programs 5-1
  - system (SYS1.LINKLIB) 5-1, 9-1
    - use of 9-2
  - SYS1.IMAGELIB
    - use of 21-11
  - temporary 5-1, 9-1
    - creating 9-4
    - use of 9-4
  - use in copy modification 24-3
- limiting, sysout output
  - requesting 25-1
  - use of 25-1
  - when exceeded 25-2
- LINDEX parameter
  - use of 24-4
- LINECT parameter
  - use of 24-2
- linect subparameter
  - use of 24-2
- LINES parameter
  - use of 7-4, 10-10, 10-14, 25-1
- lines subparameter
  - use of 25-1
- locating

- data set on tape 12-10
- log
  - See hard-copy log
  - See job log
- log subparameter
  - use of 7-7
- LOGOFF command
  - use of 6-4
- LOGON command
  - use of 6-4
- looping, program
  - stopping execution of 10-11
- LREGION parameter
  - use of 9-8

**M**

- mass storage system
  - See MSS (mass storage system)
- member
  - in PDS 9-1
- messages
  - during volume mounting 7-3
  - from system for job 7-6
  - use during testing 7-4
  - when job exceeds output limit 7-4
- mode, process
  - requesting for sysout data set 21-7
- modification
  - for sysout data set 24-3
- MODIFY parameter
  - use of 24-3
- mounting, volume
  - control of messages about 7-3
  - deferred 15-27
  - premounting 15-27
- MSGCLASS parameter
  - controlling copied input stream 6-3
  - use of 7-6, 7-7, 18-2, 21-5, 23-4
- MSGLEVEL parameter
  - use in controlling job log listing 7-2
  - use of 7-6
- MSS (mass storage system)
  - allocation of 15-19
  - nonspecific volume requests for 15-19
  - placing data sets on different volumes 15-20
  - specific volume requests for 15-20
- MSS parameter
  - use of 15-11

**N**

- naming
  - of data set 12-2
  - of temporary data set 12-3
- net
  - See dependent job net
- node

# Index

See remote node  
NOLOG parameter  
  use of 7-7  
nonstandard  
  See processing, nonstandard  
NOPWREAD subparameter  
  use of 14-2  
NOTIFY parameter  
  use of 7-5  
notifying  
  TSO userid 7-4  
null statement  
  example 4-2  
  to identify job end 4-2  
NULLFILE subparameter  
  use of 16-1  
nullifying  
  of dummy data sets 16-2  
  of dummy status for sysout data set 21-9

**O**

ONLY subparameter  
  examples 10-10  
  use of to force step execution 10-3  
ORG parameter  
  use of 23-1  
OUT subparameter  
  processing with 14-4  
  use of 14-3  
OUTLIM parameter  
  use of 23-4, 25-1  
output formatting, task for requesting sysout resources  
  chart of 24-1  
  described 24-1-24-6  
  of dumps on 3800 Printing Subsystem 24-6  
    examples 24-6  
  to any printer 24-2  
    examples 24-2  
  to punch 24-5  
    examples 24-5  
  to 3211 Printer with indexing feature 24-4  
    examples 24-5  
  to 3800 Printing Subsystem 24-3  
    examples 24-4  
OUTPUT JCL statement  
  adding parameters from 21-2  
  changing /\*OUTPUT statements to 21-4  
  changing /\*\*FORMAT statements to 21-4  
  references to multiple statements 21-2  
  use of 21-2  
output limiting, task for requesting sysout resources  
  chart of 25-1  
  described 25-1-25-2  
  examples 25-2  
  messages when limit exceeded 7-4  
  requesting 25-1  
  terminating job when limit exceeded 10-10  
  use of 25-1

  when exceeded 25-2  
OUTPUT parameter  
  use of 21-2  
overflow area  
  defined A-1  
  identifying 12-3, 12-4  
OVFLOW subparameter  
  use of 12-3

## P

PAGEDEF parameter  
  use of 21-11  
PAGES parameter  
  use of 7-4, 10-10, 10-14, 25-1  
parallel mounting, of volumes  
  to request more than one device 15-3  
parameters  
  to perform tasks 3-1  
PARM parameter  
  use in communicating from JCL to program 7-3  
  values for IBM-supplied programs 7-3  
partitioned data set  
  See PDS (partitioned data set)  
partitions  
  See spool partitions  
passing, data set  
  dismounting of volume 17-11  
  disposition when data set unreceived 17-5  
  effect on volume retention 17-11  
  receiving passed data set 17-5  
  requesting 17-5  
  when step abnormally terminates 17-5  
PASSWORD parameter  
  use of 8-1, 14-2  
passwords  
  in protecting data sets 14-2  
PDS (partitioned data set)  
  identifying 12-2, 12-4  
  members of 12-2, 12-4  
  use as library 9-1  
PEND statement  
  to identify procedure end 4-2  
PERFORM parameter  
  use of 11-5  
performance control, task for processing jobs  
  by dispatching priority 11-4  
    examples 11-4  
  by I/O-to-processing ratio 11-5  
    examples 11-5  
  by job class assignment 11-2  
    examples 11-2  
  by performance group assignment 11-5  
    examples 11-5  
  by selection priority 11-3  
    examples 11-3  
  chart of 11-1  
  described 11-1-11-5

- performance control, task for requesting sysout resources
  - by queue selection 20-1
    - example 20-1
  - chart of 20-1
  - described 20-1
- performance group
  - use of 11-5
- periodic
  - execution 5-4
- PIMSG parameter
  - use of 7-6
- postponing
  - specification of data set 16-2
- prime area
  - defined A-1
  - identifying 12-3, 12-4
- PRIME subparameter
  - use of 12-3
- Print Services Facility
  - See PSF (Print Services Facility)
- printing, sysout data set
  - controlling format of 24-2
  - on same listing 21-5
  - scheduling for 22-1
  - simultaneously on different printers 21-5
- priority
  - aging 11-4
  - dispatching 11-4
  - not useful in controlling execution order 6-2
  - selection 11-3
    - for sysout data sets 20-1
    - ignoring 20-1
    - use of 11-3
- PROC parameter
  - to execute procedure 5-2
  - use of 9-5
- PROC statement
  - to identify procedure 4-2
- procedures, cataloged and in-stream
  - defined 2-3
  - examples 2-3
  - execution of 5-2
  - overriding DD statements in 15-5
- processing control, task for processing jobs
  - by terminating execution 10-2, 10-10
    - examples when output limit exceeded 10-10
    - examples when return codes tested 10-5
  - by timing execution 10-10
    - examples 10-11, 10-12
  - chart of 10-1
  - described 10-1-10-15
  - for testing 10-12
    - by altering usual processing 10-12
    - by dumping after error 10-14
    - examples when dumping 10-15
    - examples when scanning JCL 10-13
    - examples when using IEFBR14 10-13
    - examples when using nonstandard processing 10-14
- processing control, task for requesting data set resources
  - by postponing specification 16-2
    - examples 16-3
  - by subsystem 16-5
    - example 16-5
  - by suppressing processing 16-1
    - examples 16-2
  - by TCAM job or task 16-6
    - examples 16-6
  - chart of 16-1
  - described 16-1-16-6
  - with checkpointing 16-4
    - examples 16-4
- processing control, task for requesting sysout resources
  - by checkpointing 21-10
  - by external writer 21-7
    - examples 21-7
  - by holding 21-8
    - examples 21-9
  - by mode 21-7
    - examples 21-7
  - by PSF 21-11
    - examples 21-11
  - by suppressing output 21-9
    - examples 21-9, 21-10
  - chart of 21-1
  - described 21-1-21-11
  - with additional parameters 21-2
    - examples 21-2
  - with checkpointing
    - examples 21-11
  - with other data sets 21-5
    - examples 21-5, 21-6
- processing jobs
  - task in job control 2-4
  - tasks
    - chart of 3-5
- processing, nonstandard
  - defined 10-14
  - use in testing 10-14
- processor
  - as output destination 23-4
  - selecting in JES2 9-8
  - selecting in JES3 9-9
- PROCLIB parameter
  - use of 9-5
- programmer's name
  - to identify 4-4
- PROTECT parameter
  - use of 14-1
- protection, task for entering jobs
  - chart of 8-1
  - described 8-1
  - through RACF 8-1
    - examples 8-1
- protection, task for requesting data set resources
  - by passwords 14-2
    - examples 14-3
  - chart of 14-1
  - described 14-1-14-4

# Index

- for ISO/ANSI/FIPS Version 3 tapes 14-2
  - examples 14-2
- of access to BSAM and BDAM data sets 14-3
  - chart of 14-3
  - examples 14-4
- through RACF 14-1
  - examples 14-2
- PRTY parameter
  - use of 11-3, 20-1
- PSF (Print Services Facility)
  - controlling 21-11
- punching, sysout data set
  - formatting of 24-5
  - interpretation 24-5
  - scheduling for 22-1

**Q**

- QNAME parameter
  - use of 12-10, 16-6

**R**

- RACF (resource access control facility)
  - data set protection 14-1
  - protection through 8-1
- RD parameter
  - in restarts 5-3
- reader, input
  - holding 6-2
- reader, internal
  - as output destination 23-4
  - limiting records to 23-4
  - message class for 23-4
  - sending directly to JES 23-4
- receiving, passed data set
  - requesting 17-5
- relative generation numbers
  - defined B-1
- releasing, held sysout data set
  - printing 21-8
  - requesting 21-8
- remote job entry
  - See RJE (remote job entry)
- remote job processing
  - See RJP (remote job processing)
- remote node
  - execution at 5-7
  - specifying as destination for sysout data set 23-1
- remote terminal
  - use of 6-4
- remote work station
  - use of 6-4
- requesting resources
  - for data sets 2-4
  - for sysout data sets 2-4
  - task in job control 2-4
  - tasks

- chart of 3-6, 3-9
- resource access control facility
  - See RACF (resource access control facility)
- resource control, task for entering jobs
  - chart of 9-1
  - described 9-1-9-11
  - of address space 9-6
    - examples 9-8
  - of procedure library 9-5
    - examples 9-5
  - of processor 9-8
    - examples 9-9, 9-10
  - of program library 9-1
    - example of concatenating 9-4
    - example of creating and adding to 9-3
    - example of retrieving 9-3
    - example of temporary 9-4
  - of spool partition 9-10
    - examples 9-11
- RESTART parameter
  - in restarts 5-3
- restarting
  - after abnormal termination 5-2
  - after system failure (JES2 system) 5-3
  - after system failure (JES3 system) 5-3
  - automatic checkpoint 5-2
  - automatic step 5-2
  - deferred checkpoint 5-2
  - deferred step 5-2
  - use of 5-3
  - when job contains generation data sets B-5
- RETAIN subparameter
  - use of 17-10, 17-11
- retention, of tape volume
  - demounting of volume 17-11
  - requesting 17-11
  - use of 17-11
- RETPD subparameter
  - use of 17-8
- return code
  - compatible tests 10-4
  - examples 10-5
  - use of 10-2
- RJE (remote job entry)
  - use of 5-7, 6-3
- RJP (remote job processing)
  - output destinations for 23-3
  - use of 5-7, 6-4

**S**

- scanning
  - syntax for errors 6-3, 10-12
- scratch disposition
  - for tape volume 17-10
- SETUP parameter
  - mount messages for volumes 7-3
  - use of 15-11
- setup, of devices

- altering 15-14
  - explicit 15-13
  - high watermark 15-12
  - in JES3 system 15-12
  - job 15-12
  - SMF (System Management Facilities)
    - to establish exit routine when execution time exceeded 10-11
  - SNA/SDLC (systems network architecture synchronous data link control)
    - use of 6-3, 6-4
  - SPACE parameter
    - use of 15-21, 17-9, A-4, B-3
  - space, requesting
    - blocks 15-21
    - cylinders 15-21
    - for indexed sequential data sets 15-23
    - for PDS directory 15-23
    - primary
      - how system allocates 15-21
    - releasing when unused 17-9
    - secondary
      - for NEW or MOD data set 15-22
      - for OLD data set 15-22
      - how system allocates 15-22
      - in blocks 15-23
      - only for current execution 15-23
    - specific tracks 15-24
    - system assignment 15-21
    - tracks 15-21
    - with user labels 15-22, 15-24
  - SPART parameter
    - use of 9-11
  - spinning off, sysout data sets
    - requesting 22-1
    - use of 22-1
  - spool partitions
    - controlling allocation of 9-10
  - statements
    - See also JCL, JES2, and JES3
    - purposes of 1-1-1-2
    - to perform tasks 3-1
  - station
    - See remote work station
  - status
    - of data set
      - specifying 13-1
    - of device
      - affect on allocation 15-2
  - STPCAT catalog
    - use of 12-7
  - STEPLIB
    - See libraries, private (JOBLIB or STEPLIB)
  - stepname
    - to identify step 4-2
  - steps, job
    - defined 2-1
    - examples 2-1
    - number of 2-4
  - storage
    - logical 9-8
    - real 9-6
      - region size for 9-7
    - requesting 9-6
    - virtual 9-6
      - region size for 9-7
  - SUBSYS parameter
    - use of 16-5
  - subsystem
    - printing messages from 7-6
    - program control statements for 16-5
    - requesting 16-5
  - suppressing, sysout output
    - requesting 21-9
    - use of with started tasks 21-10
  - synchronous data link control
    - See SNA/SDLC (systems network architecture synchronous data link control)
  - syntax
    - scanning for errors 6-3, 10-12
  - SYSABEND statement
    - use of 10-14
  - SYSAFF parameter
    - use of 9-8
  - SYSCHK DD statement
    - in restarts 5-3
  - SYSCKEOV DD statement
    - use of 21-10
  - SYSMDUMP statement
    - use of 10-14
  - sysout data set
    - printing with job log 7-7
  - SYSOUT parameter
    - use of 7-7, 18-1, 18-2
  - System Management Facilities
    - See SMF (System Management Facilities)
  - SYSTEM parameter
    - use of 9-9
  - system-generated qualified name
    - for temporary data set 12-3
  - systems network architecture
    - See SNA/SDLC (systems network architecture synchronous data link control)
  - SYSUDUMP statement
    - use of 10-14
  - SYS1.LINKLIB
    - See libraries, system (SYS1.LINKLIB)
  - SYS1.PROCLIB
    - See catalogs, of job control procedures (SYS1.PROCLIB)
- T
- tasks
    - See also tasks by name
    - charts for 3-1
    - for job control iii
    - required 3-1
  - TCAM (telecommunications access method)

# Index

message data set 12-10  
processing of TCAM message data set 16-6  
telecommunications access method  
See TCAM (telecommunications access method)  
TERM parameter  
use of 12-11, 23-6  
terminal  
See also remote terminal  
as output destination 23-6  
identifying data sets from or to 12-11  
termination, abnormal  
because execution time exceeded 10-11  
data set disposition during 17-2  
disposition of unreceived passed data sets 17-5  
during allocation, effect on disposition 17-2  
during execution, effect on disposition 17-2  
effect on passing of data set 17-5  
forcing execution of later step 10-3  
restarting after 5-2  
when output limit exceeded 10-10  
termination, normal  
data set disposition during 17-2  
THRESHLD parameter  
use of 21-5  
TIME parameter  
use of 10-11, 10-12  
time sharing option  
See TSO (time sharing option) userid  
TRC parameter  
use of 24-3, 24-4  
TSO (time sharing option) userid  
as output destination 23-4  
notifying when job completes 7-4  
RACF protection parameters from logon 8-1  
TYPE parameter  
when requesting processor 9-10  
TYPRUN parameter  
for copying job 6-3  
in holding job 6-1  
use of 10-12

U

UCS parameter  
use of 24-2, 24-4  
uncataloging, data set  
of generation data set B-5  
requesting 17-4  
unit  
See device  
UNIT parameter  
definition during system generation 15-3  
for output data set 15-8  
relationship to VOLUME parameter 15-5  
use of 15-1, A-2, A-6, B-3, B-4  
when requesting processor 9-10  
unreceived data set  
at abnormal termination 17-5  
at end of job 17-6

disposition of 17-5  
UPDATE parameter  
in holding job 6-2  
use of 9-5  
USER parameter  
use in identifying job with TSO userid 7-5  
use of 8-1

## V

VIO (virtual input/output)  
backward references to 15-25  
use of 15-24  
virtual input/output  
See VIO (virtual input/output)  
virtual storage access method  
See VSAM (virtual storage access method) data sets  
volume attributes  
affect on device allocation 15-8  
defined 15-15  
permanently resident 15-15  
private 15-15  
assigning attribute 15-17  
retention of 17-11  
use of 15-17  
public 15-15  
assigning attribute 15-17  
retention of 17-11  
removable 15-15  
reserved 15-15  
storage 15-15  
VOLUME parameter  
references to, in earlier DD statement 15-6  
relationship to UNIT parameter 15-5  
use of 15-15, 15-16, A-2, A-6, B-3, B-4  
volume requests  
for mass storage groups 15-19  
nonspecific 15-16  
for MSS 15-19  
how system allocates 15-16  
label type for 12-9  
number per DD statement 15-18  
specific 15-15  
for MSS 15-20  
how system allocates 15-16  
label type for 12-9  
VSAM (virtual storage access method) data sets  
creating C-1  
described C-1-C-4  
parameters to avoid C-2  
parameters used with C-2  
retrieving C-1

W

WARNING subparameter  
use of 25-2  
work station



- See remote work station
- WRITER parameter
  - use of 21-7
- writer, external
  - for processing sysout data set 21-7
  - requesting 21-7
- 3
- 3203 Printer Model 5
  - for printing sysout data set 24-2
- 3211 Printer
  - for printing sysout data set 24-4
- 3450 Diskette Input/Output Unit
  - identifying output data set for 18-2
  - input data sets on 12-6
- 3800 Printing Subsystem
  - for printing high-density dumps 24-6
  - for printing sysout data set 21-11, 24-3
- 3850 Mass Storage System
  - allocation of 15-19

# Index





Printed in U.S.A.

This manual is part of a library that serves as a reference source for systems 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.

**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.)

Note: Staples can cause problems with automated mail sorting equipment.  
Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

Reader's Comment Form

Cut or Fold Along Line

Fold and tape

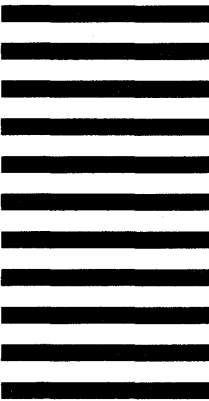
Please Do Not Staple

Fold and tape



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.



POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation  
Department D58, Building 921-2  
PO Box 390  
Poughkeepsie, New York 12602

Fold and tape

Please Do Not Staple

Fold and tape

Printed in U.S.A.



This manual is part of a library that serves as a reference source for systems 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.

**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:

---

---

---

---

Note: Staples can cause problems with automated mail sorting equipment.  
Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

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

Cut or Fold Along Line

Fold and tape

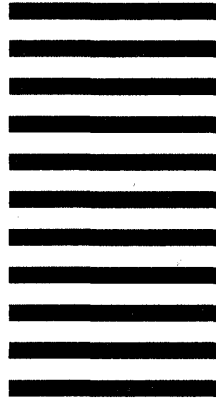
Please Do Not Staple

Fold and tape



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.



POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation  
Department D58, Building 921-2  
PO Box 390  
Poughkeepsie, New York 12602

Fold and tape

Please Do Not Staple

Fold and tape

Printed in U.S.A.



GC28-1351-00

