GC28-1352-4
File No. S370-36

Program Product

MVS/Extended Architecture JCL Reference

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 2.3 (program number 5740-XC6)**
- **MVS/System Product - JES3 Version 2 Release 2.3 (program number 5665-291)**
- **MVS/Extended Architecture Data Facility Product (DFP) Version 2 Release 3.0 (program number 5665-XA2) and MVS/Data Facility Product (MVS/DFP) Version 3 Release 1.0 (program number 5665-XA3)**
- **Resource Access Control Facility (RACF) Version 1 Release 7 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.

Fifth Edition (September, 1989)

This is a major revision of, and obsoletes, GC28-1352-3 and Technical Newsletters GN28-1198 and GN28-1254. See the Summary of Amendments following the Contents for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to the program releases listed in the box above and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters.

The book, *MVS/Extended Architecture JCL Reference*, GC28-1352-2, applies to MVS/System Product Version 2 Releases 1.2, 1.3, 1.5, and 1.7 and may now be ordered using the temporary order number GQ28-1352. 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 or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product in this publication is not intended to state or imply that only IBM's product may be used. Any functionally equivalent product may be used instead. This statement does not expressly or implicitly waive any intellectual property right IBM may hold in any product mentioned herein.

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 950, 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 lists 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 code 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 reference book, to be used while coding the statements. It contains some introductory material. Full explanations of the job control tasks are presented in a companion book, *MVS/Extended Architecture JCL User's Guide*, GC28-1351.

Trademarks

The following are trademarks of International Business Machines Corporation.

- MVS/DFP™
- MVS/SP™
- MVS/XA™

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

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

Chapter 2. Job Control Tasks: This chapter contains charts of job control tasks and the statements and parameters that can be used to perform the tasks.

Chapter 3. Format of Job Control Statements: This chapter describes the fields in job control statements and how to continue fields onto following statements.

Chapter 4. Syntax: This chapter describes:

- Notation used to show the syntax of job control parameters.
- Character sets used in coding job control statements.
- Backward references and their use.

Chapter 5. Cataloged and In-Stream Procedures: This chapter presents the information needed to code and use cataloged and in-stream procedures.

Chapter 6. Job Control Statements on the Output Listing: This chapter shows how job control statements are identified on the job log output listing.

Chapters 7 through 20: These chapters detail the coding of each JCL statement, in alphabetical order, and of each parameter, in alphabetical order by statement. The chapters are:

Chapter 7. Command Statement
Chapter 8. Comment Statement
Chapter 9. CNTL Statement
Chapter 10. DD Statement
Chapter 11. Special DD Statements
Chapter 12. Delimiter Statement
Chapter 13. ENDCNTL Statement
Chapter 14. EXEC Statement
Chapter 15. JOB Statement
Chapter 16. Null Statement
Chapter 17. OUTPUT JCL Statement
Chapter 18. PEND Statement
Chapter 19. PROC Statement
Chapter 20. XMIT JCL Statement

For the DD, EXEC, JOB, and OUTPUT JCL statements, the chapter introduction includes charts summarizing all the parameters, their values, and their purposes. These charts should help the new JCL coder select parameters and act as a reminder for the experienced coder.

Except for chapter 11, each of these chapters cover only one statement. In them, the parameters are listed alphabetically. For each statement and parameter, the following information is given, as needed:

- **Parameter type**: positional or keyword, required or optional, and if it applies for SMS.
- **Purpose** of the parameter.
- **References** to related information in other IBM publications.
- **Syntax** and coding rules.
- **Parameter** or **subparameter definitions**: how to code each parameter or subparameter.
- **Defaults** if you do not code a statement, parameter, or subparameter.
- **Overrides**: statements that this statement overrides or is overridden by or parameters that this parameter overrides or is overridden by.

- **Relationship to other parameters**, including other parameters or subparameters that should not be coded with this one.
- **Relationship to other control statements**.
- **On EXEC statement that calls a procedure**, for EXEC statement parameters.
- **Location in the JCL**.
- Other information required to code the statement or parameter.
- **Examples**.

Chapter 21. JES2 Statements: This chapter details the coding of JES2 control statements.

Chapter 22. JES3 Statements: This chapter details the coding of JES3 control statements.

Guide to Using this Publication

If your system contains MVS/System Product - JES2 Version 2 (program number 5740-XC6), JES2 information in this manual refers to the JES2 function in the MVS/System Product Version 2, unless otherwise noted.

If your system contains MVS/System Product - JES3 Version 2 (program number 5665-291), JES3 information in this manual refers to the JES3 function in the MVS/System Product Version 2, unless otherwise noted.

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

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

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

JCL DD parameters supported differently are:

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

Supported in MVS/System Product Version 1 but not in Version 2 are the CODE and FRID subparameters of the DCB parameter on the DD statement.

Prerequisite Publication

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

Co-requisite Publication

MVS/Extended Architecture JCL User's Guide, GC28-1351.

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 6, LY28-1176 through LY28-1181.
MVS/Extended Architecture Installation: System Generation, GC26-4148.
MVS/Extended Architecture Interactive Problem Control System (IPCS) User's Guide, GC28-1407.

Data Facility Product - Version 2

MVS/Extended Architecture System-Data Administration, GC26-4149.
MVS/Extended Architecture Data Administration Guide, GC26-4140.
MVS/Extended Architecture Integrated Catalog Administration: Access Method Services Reference, GC26-4135.
MVS/Extended Architecture VSAM Catalog Administration: Access Method Services Reference, GC26-4136.
MVS/Extended Architecture Checkpoint/Restart User's Guide, GC26-4139.
MVS/Extended Architecture Data Administration: Utilities, GC26-4150.
MVS/Extended Architecture Magnetic Tape Labels and File Structure Administration, GC26-4145.
MVS/Extended Architecture VSAM Administration Guide, GC26-4151.
MVS/Extended Architecture Interactive Storage Management Facility User's Guide, GC26-4266.

Data Facility Product - Version 3

MVS/ESA System-Data Administration, SC26-4515.
MVS/ESA Data Administration Guide, SC26-4505.
MVS/ESA Storage Administration Reference, SC26-4514.
MVS/ESA Integrated Catalog Administration: Access Method Services Reference, SC26-4500.
MVS/ESA VSAM Catalog Administration: Access Method Services Reference, SC26-4501.
MVS/ESA Checkpoint/Restart User's Guide, SC26-4503.
MVS/ESA Data Administration: Utilities, SC26-4516.
MVS/ESA Magnetic Tape Labels and File Structure Administration, SC26-4511.
MVS/ESA VSAM Administration Guide, SC26-4518.
MVS/ESA Interactive Storage Management Facility User's Guide, SC26-4508.

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

Advanced Communications Function for VTAM Version 2 Programming, SC27-0611.
Advanced Communications Function for TCAM, Version 2 Installation Reference, SC30-3133.
OS/VS Mass Storage System (MSS) Services General Information, GC35-0016.
Resource Access Control Facility (RACF) General Information Manual, GC28-0722.
Resource Access Control Facility (RACF) Security Administrator's Guide, SC28-1340.
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.
Print Services Facility User's Programming Guide for MVS, S544-3084.
Print Services Facility System Programmer's Guide for MVS, SH35-0091.

Contents

Figures

Summary of Amendments

Summary of Amendments
for GC28-1352-4
for MVS/System Product Version 2 Release 2.3

This major revision supports MVS/System Product Version 2 Release 2.3. Changes include:

- MVS/XA support for MVS/Data Facility Product (MVS/DFP) Version 3 Release 1.0, which introduces the Storage Management Subsystem (SMS). SMS provides new function for data and storage management.

  In this book, "with SMS" indicates information that applies when SMS is installed and active; "without SMS" indicates SMS is not installed or is not active.

  For JCL, MVS/XA adds nine DD statement parameters for defining new data sets with SMS. Note that the system ignores these DD parameters when SMS is not installed or is not active.

  The new DD parameters are:

  - AVGREC - average record
  - DATACLAS - data class
  - KEYOFF - key offset
  - LIKE - like dsname
  - MGMTCLAS - management class
  - RECORG - record organization
  - REFDD - reference ddname
  - SECMODEL - security model
  - STORCLAS - storage class *

    * When a storage class is assigned to a new data set, the data set is referred to as an "SMS-managed data set".

  There are also changes to the following DD statement parameters for defining data sets with SMS.

  - SPACE=(reclgth... - record length
  - SPACE=(,(,,directory)) - directory quantity only

  With SMS, do not use the JOBCAT DD statement in jobs and STEPCAT DD statements in steps that reference SMS-managed data sets. SMS only accesses SMS-managed data sets that are cataloged in a system catalog.

With SMS, you can define new VSAM data sets with DD statements (see the DATACLAS and RECORG parameters). Also, the disposition processing of VSAM data sets is changed (see the DISP parameter).

- Alternate syntax for the DD LABEL and DCB parameters.
  - On the LABEL DD parameter, you can specify EXPDT and RETPD subparameters without the need to code LABEL=.
  - On the DCB DD parameter, you can specify all DCB keyword subparameters without the need to code DCB=.
- The following subparameters of the LABEL and DCB DD parameters are now described as DD parameters.
  - EXPDT - expiration date (from LABEL)
  - RETPD - retention period (from LABEL)
  - KEYLEN - key length (from DCB)
  - LRECL - logical record length (from DCB)
  - RECFM - record format (from DCB)
- Maintenance updates, including the following:
  - Parameter THWSSEP is added to the //*MAIN statement for JES3 support of the IBM 3480 Automatic Cartridge Loader (from Technical Newsletter GN28-1198).
  - Parameter SUBCHARS is added to the // XMIT JCL statement (from Technical Newsletter GN28-1254).
  - On the JES3 //*MAIN statement, the mmm subparameter is added for the WARNING subparameter on the BYTES, CARDS, LINES, and PAGES parameters.
  - Information for RACF Release 1.6 is removed from the DD PROTECT parameter.
  - Information for previous releases of JES2 related to held data sets is removed from the DD SYSOUT and OUTPUT JCL CLASS parameters.
  - The reference tables in Chapter 23 are removed.
    - For mutually exclusive DD parameters, see the description of the individual parameters on the DD statement.
    - For direct access and track capacities, see *Data Administration Guide*.
  - Change to DCB BLKSIZE. If BLKSIZE is not specified, the system determines an optimum block size for the data.
  - Updates are made to the DD AMP parameter.
  - Other updates throughout the book.

Summary of Amendments
for GC28-1352-3
as Revised June, 1987

This major revision supports MVS/System Product Version 2 Release 2.0. Changes include:

- An expiration date with a four-digit year (through 2155) can be specified for a data set on the EXPDT subparameter of the LABEL parameter.
- The expiration date for a data set that is calculated with the RETPD subparameter of the LABEL parameter uses 366 days for leap years.
- The maximum number of DD statements allowed per job step is increased from 1635 to 3273.
- Maintenance updates, including the following:
  - Information for the JES2 Greater than 99 Printer Support Enhancement, which allows route codes up to 9999 to be specified for remote work stations and local and remote terminals.
  - Information for the DATACK and PIMSG parameters on the OUTPUT JCL statement.
  - Other changes throughout the book.

Summary of Amendments
for GC28-1352-2
as Revised July 31, 1986

This major revision supports the JES3 SNA/NJE Enhancement by addition of the XMIT JCL statement.

It also contains maintenance updates.

Chapter 1. Job Control Statements

This chapter lists in Figure 1-1 all the job control statements and gives the purpose of each statement. Each statement is described in detail in later chapters.

Statement	Name	Purpose
JCL Statements		
// command	JCL command	Enters an MVS system operator command through the input stream. The command statement is used primarily by the operator. **Note:** 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. **Note:** 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.
// XMIT	transmit	Transmits input stream records from one node to another. **Note:** The XMIT JCL statement is supported only on JES3 systems.

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

Statement	Purpose
JES2 Control Statements	
/*$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.
JES3 Control Statements	
//**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 Tasks

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 an MVS/SP base control program (BCP) with a job entry subsystem (JES2 or JES3) and the Data Facility Product (DFP) 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 consist of:

JCL statements
JES2 control statements
JES3 control statements

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.

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 though it were an 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 (or an installation-defined procedure library), 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.

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 a JCL error.

Tasks

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.

Task Charts

The following charts list the job control tasks, which are described in the *JCL User's Guide*, in four groups:

- Entering jobs in Figure 2-1 on page 2-3
- Processing jobs in Figure 2-2 on page 2-6
- Requesting data set resources in Figure 2-3 on page 2-7
- Requesting sysout data set resources in Figure 2-4 on page 2-10

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 FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Identification					
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
Execution					
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			XMIT JCL (JES3 only)	/*ROUTE XEQ /*XEQ /*XMIT	//*ROUTE XEQ

Figure 2-1 (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		
Job input control					
by holding job entrance	TYPRUN CLASS				HOLD, UPDATE, or CLASS on //*MAIN //*NET
by holding local input reader					//*PAUSE
by copying input stream (JES2 only)	TYPRUN CLASS				
from remote work station				/*SIGNON /*SIGNOFF	/*SIGNON /*SIGNOFF
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 2-1 (Part 2 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		
Protection					
through RACF	GROUP PASSWORD USER				
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 2-1 (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		
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	TYPRUN CLASS	PGM=IEFBR14 PGM=JCLTEST PGM=JSTTEST (JES3 only)			//*PROCESS //*ENDPROCESS
(2) by dumping after error			SYSABEND DD SYSMDUMP DD SYSUDUMP DD To format dump on 3800 Printing Subsystem, FCB=STD3 and CHARS=DUMP on dump DD		DUMP in BYTES, CARDS, LINES, and PAGES on //*MAIN
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 2-2. Tasks for Processing Jobs

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Identification					
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/Output Unit	DSID				
through catalog	JOBCAT DD STEPCAT 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				
Description					
of status	DISP				
of data attributes - by modeling	DCB AMP DATACLAS KEYLEN KEYOFF LRECL RECFM RECORG LIKE REFDD				
of migration and backup	MGMTCLAS				

Figure 2-3 (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		
Protection					
through RACF	PROTECT SECMODEL				
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				
Allocation					
of device	UNIT STORCLAS		CLASS on JOB (JES3 only)		SETUP, MSS, and CLASS on //*MAIN
of tape or direct access volume	VOLUME MSVGP STORCLAS				EXPDTCHK and RINGCHK on //*MAIN
of direct access space	SPACE AVGREC DATACLAS				
of virtual I/O	UNIT DSNAME = temporary data set				
with deferred volume mounting	DEFER on UNIT				
with volume premounting				/*SETUP	
dynamic			DYNAMNBR on EXEC		

Figure 2-3 (Part 2 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		
Processing control					
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				
End processing					
deallocation	FREE				
disposition of data set	DISP RETPD EXPDT				
release of unused direct access space	RLSE on SPACE				
disposition of volume	RETAIN and PRIVATE on VOLUME				

Figure 2-3 (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		
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				
Description					
of data attributes	DCB				
Performance control					
by queue selection		PRTY			

Figure 2-4 (Part 1 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		
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	CLASS			
by suppressing output	DUMMY class on SYSOUT				
with checkpointing		CKPTLINE CKPTPAGE CKPTSEC			
by Print Services Facility (PSF)		FORMDEF PAGEDEF			

Figure 2-4 (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 2-4 (Part 3 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		
Output formatting					
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 Subsystem, 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 Subsystem	CHARS=DUMP FCB=STD3	CHARS=DUMP FCB=STD3			
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 2-4 (Part 4 of 4). Tasks for Requesting Sysout Data Set Resources

Chapter 3. Format of Statements

This chapter describes the fields in JCL, JES2 and JES3 statements. It ends with the conventions for continuing statements.

JCL Statement Fields

A JCL statement consists of one or more 80-byte records. Each record is in the form of an 80-column punched-card image. Each JCL statement is logically divided into the following five fields. All five fields do not appear on every statement; see Figure 3-1 on page 3-2 for the fields that can appear on each statement.

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

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

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

- The name must begin in column 3.
- The name is 1 through 8 alphanumeric or national ($, #, @) characters. See Figure 4-2 on page 4-4 for the character sets.
- The first character must be an alphabetic or national ($, #, @).
- The name must be followed by at least one blank.

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

- The operation field consists of the characters in the syntax box for the statement.
- The operation follows the name field.
- The operation must be preceded and followed by at least one blank.

Parameter field

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

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

See "Parameter Field" on page 3-3 for details on coding the parameter field.

Comments field

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

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

You can code comments after the parameter field even though you continue the parameter field on a subsequent statement; see "Continuing JCL Statements" on page 3-5.

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

Statement	Fields
JCL Command	// command [parameter] [comments]
Comment	//* comments
CNTL	//label CNTL [* comments]
DD	//[ddname] DD [parameter [comments]] //[ddname] DD
Delimiter	/* [comments] xx [comments]
ENDCNTL	//[label] ENDCNTL [comments]
EXEC	//[stepname] EXEC parameter [comments]
JOB	//jobname JOB [parameter [comments]] //jobname JOB
Null	//
OUTPUT JCL	//name OUTPUT parameter [comments]
PEND	//[name] PEND [comments]
PROC (cataloged)	//[name] PROC [parameter [comments]] //[name] PROC
PROC (in-stream)	//name PROC [parameter [comments]] //name PROC

Figure 3-1. JCL Statement Fields

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

Do not code fields, except on the comment statement, past column 71. If the total length of the fields would exceed 71 columns, continue the fields onto one or more following statements. Continuing fields is described under "Continuing JCL Statements" on page 3-5. The comment statement can be coded through column 80.

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

Parameter Field

The parameter field consists of two types of parameters: **positional parameters** and **keyword parameters**. All positional parameters must precede all keyword parameters. Keyword parameters follow the positional parameters.

Commas: Use commas to separate positional parameters, keyword parameters, and subparameters in the parameter field.

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

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

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

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

Code any of the keyword parameters for a statement in any order in the parameter field after the positional parameters. Because of this positional independence, never code a comma to indicate the absence of a keyword parameter.

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

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

JES2 Control Statement Fields

The rules for coding JES2 control statements are the same as the rules for JCL statements, with the following additions:

- Columns 1 and 2 always contain the characters /*
- Do not code comments on any JES2 statements. Where comments are needed, code a JCL comment statement.
- If you code the same parameter on the same statement more than once, JES2 uses the value in the last parameter.

When coding a JES2 control statement more than once, be aware of the following JES2 actions:

- If the same parameter appears on more than one statement, JES2 uses the value coded on the last statement.
- If the statements contain different parameters, JES2 uses all parameters combined.

JES3 Control Statement Fields

The rules for coding JES3 control statements are the same as the rules for JCL statements, with the following additions:

- Columns 1, 2, and 3 contain the characters //* except for two JES3 control statements that contain /* in columns 1 and 2.
- Columns 3 and 4 must not be blank.
- Do not code comments on JES3 control statements, except //*ENDPROCESS and //*PAUSE statements. Where additional comments are needed, code a JCL comment statement.

Continuing Statements

Continuing JCL Statements

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

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

Command statement
Comment statement
Delimiter statement
Null statement

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

Continuing the Parameter Field:

1. Interrupt the field after a complete parameter or subparameter, including the comma that follows it, at or before column 71.

2. Include comments by following the interrupted parameter field with at least one blank.

3. Code a nonblank character in column 72 when you are continuing a comments field and, optionally, when you are continuing the parameter field.

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

4. Code // in columns 1 and 2 of the following statement.

5. Continue the interrupted parameter or field beginning in any column from 4 through 16. If you begin coding after column 16, the system treats this statement as a comment field.

6. Column 3 of the following statement can contain only a blank or an asterisk. If column 3 contains anything else, the system assumes the following statement is a new statement. The system issues an error message indicating that no continuation is found and fails the job.

Continuing the Comments Field:

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

You can also use a comments statement to include continued comments.

Format: Continuing Statements

Examples of Continued Statements:

```
//DS1 DD  DSNAME=INDS,DISP=OLD,CHKPT=EOV,   MY INPUT DATA SET
//     UNIT=SYSSQ,VOLUME=SER=(TAPE01,TAPE02,TAPE03)
```

```
//DD1  DD   DSNAME=SWITCH.LEVEL18.GROUP12,UNIT=3350,
//     VOLUME=335023,SPACE=(TRK,(80,15)),DISP=(,PASS)
```

```
//STP4  EXEC PROC=BILLING,COND.PAID=((20,LT),EVEN),
//     COND.LATE=(60,GT,FIND),
//     COND.BILL=((20,GE),(30,LT,CHGE))  THIS STATEMENT CALLS        X
//     THE BILLING PROCEDURE AND SPECIFIES RETURN CODE TESTS
//*    FOR THREE PROCEDURE STEPS.
```

This example shows continuation of comments, first, according to the four rules above and, then, onto a comments statement.

Continuing JES2 Control Statements

The only JES2 control statement that you can continue is the /*OUTPUT statement. For all other JES2 control statements, code the statement as many times as needed.

Continuing JES3 Control Statements

Continue JES3 statements, except the command statement or //*NETACCT statement, by:

1. Coding a comma as the last character of the first statement.
2. Coding //* in columns 1 through 3 of the continuation statement.
3. Resuming the code in column 4 of the continuation statement.

On the JES3 //*NET statement, each parameter must appear entirely on one statement; a subparameter cannot be continued after a comma, except for the RELEASE parameter. To continue the RELEASE parameter, end the statement with the comma following a jobname and continue the next statement with the next jobname. The left parenthesis appears at the beginning of the jobname list and the right parenthesis appears at the end of the list. For example:

```
//*NET NETID=EXP1,RELEASE=(JOB35,JOB27Z,MYJOB,
//*WRITJB,JOBABC)
```

If the parameters on a //*NETACCT statement cannot fit on one statement, code more than one //*NETACCT statement.

Chapter 4. Syntax of Parameters

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

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

The syntax rules apply to all job control statements: JCL statements, JES2 control statements, and JES3 control statements.

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

Notation Used to Show Syntax

The syntax of the job control statements and of their parameters appear in the chapters that describe the statements. The notation used in this publication for the syntax is shown in Figure 4-1 on page 4-2.

Syntax: Notation

Notation	Meaning	Examples
Uppercase letters, words, and characters	Code uppercase letters, words, and the following characters exactly as they appear in the syntax. & ampersand * asterisk , comma = equal sign ( ) parentheses . period / slash	
Lowercase letters, words, and symbols	Lowercase letters, words, and symbols in the syntax represent variables. Substitute specific information for them.	Syntax: on JOB statement CLASS=jobname Coded: CLASS=A
| (vertical bar)	A vertical bar indicates an exclusive OR. Never code | on a control statement. It is used between choices within braces or brackets; it indicates that you code only one of the items within the braces or brackets.	Syntax: on DD DCB parameter BFALN={F|D} Coded: BFALN=F or BFALN=D
{ } (braces)	Braces surround *required*, related items and indicate that you must code one of the enclosed items. Never code { or } on a control statement.	Syntax: on DD SPACE parameter {TRK {CYL {blklgth} Coded: TRK CYL 960
[ ] (brackets)	Brackets surround an *optional* item or items and indicate that you can code one or none of the enclosed items. Never code [or] on a control statement.	Syntax: on DD UNIT parameter [,DEFER] Coded: ,DEFER or omitted Syntax: on DD LABEL parameter [,RETPD=nnnn ,EXPDT={yyddd {yyyy/ddd}] Coded: ,RETPD=nnnn or ,EXPDT=yyddd or ,EXPDT=yyyy/ddd or omitted

Figure 4-1 (Part 1 of 2). Notation Used to Show Syntax

Notation	Meaning	Examples
{ , } or [,]	One of the items in braces or brackets can be a comma. Code the comma when you do not code any of the other items in the braces or brackets but you are coding a following part of the parameter.	Syntax: on DD UCS parameter UCS=(character-set-code[,FOLD|,] [,VERIFY]) Coded: UCS=(character-set-code) UCS=(character-set-code,FOLD) UCS=(character-set-code,FOLD,VERIFY) UCS=(character-set-code,,VERIFY) Note that the comma is not coded if both FOLD and VERIFY are omitted, but must appear if FOLD is omitted and VERIFY follows.
_ (underline)	An underline indicates the default that the system uses when you do not code a subparameter.	Syntax: on JOB or EXEC statement ADDRSPC={VIRT|REAL} Coded: ADDRSPC omitted means ADDRSPC=VIRT
... (ellipsis)	An ellipsis follows an item that you can code more than once. Never code ... on a control statement.	Syntax: on DD statement COND=((code,operator)[,(code,operator)]....) Coded: Can repeat ,(code,operator) Thus: COND=((12,GE),(8,EQ),(4,EQ))
.. (two consecutive periods)	Two consecutive periods indicate that a parameter consists of a symbolic parameter followed by a period and then by other code, so that only part of the parameter is variable.	Coded: &DEPT..NYC Meaning: If &DEPT is D27: D27.NYC is the value

Figure 4-1 (Part 2 of 2). Notation Used to Show Syntax

Character Sets

To code job control statements, use characters from the three character sets in Figure 4-2. Figure 4-3 lists the special characters that have syntactical functions in job control statements.

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

Figure 4-2. Character Sets

Character	Syntactical Function
,	To separate parameters and subparameters
=	To separate a keyword from its value, for example, BURST=YES
( )	To enclose a parameter or subparameter list
&	To identify a symbolic parameter, for example, &LIB
&&	To identify a temporary data set name, for example, &&TEMPDS
.	To separate parts of a qualified data set name, for example, A.B.C., or parts of certain parameters or subparameters, for example, nodename.userid

*	To refer to an earlier statement, for example, OUTPUT=*.name, or, in certain statements, to indicate special functions:
	//label CNTL *
	//ddname DD *
	RESTART=* on the JOB statement

(blank)	To delimit fields

Figure 4-3. Special Characters Used in Syntax

Special Characters in Parameters: The syntax or parameter description indicates if the variable that you code can contain special characters or not. Parameters and subparameters that can contain special characters not used for syntactical functions usually must be enclosed in apostrophes, for example, ACCT='123+456'. Code each apostrophe that is part of the parameter or subparameter as two consecutive apostrophes, for example, code O'Neil as 'O''NEIL'.

Warning: A blank can be coded only in the following:

Accounting information on the JOB statement
Programmer's name on the JOB statement
PARM parameter on the EXEC statement
DSNAME parameter on the DD statement

When so used, the parameter or subparameter must be enclosed in apostrophes. In any other parameter, even though enclosed in apostrophes, a blank is interpreted as the end of the parameter field; the system regards anything that follows as comments.

Figure 4-4 lists the parameters that can contain certain special characters without requiring enclosing apostrophes.

Statement and Parameter or Subparameter	Special Characters Not Needing Enclosing Apostrophes	Examples
JOB accounting information	Hyphens (-)	//JOBA JOB D58-D04

JOB programmer's-name	Hyphens (-), leading periods, or embedded periods. Note that a trailing period requires enclosing apostrophes.	//JOBB JOB ,S-M-TU
		//JOBC JOB ,.ABC
		//JOBD JOB ,P.F.M
		//JOBE JOB ,'A.B.C.'

EXEC ACCT	Hyphens (-) or plus zero (+0, an overpunch)	//S1 EXEC PGM=A,ACCT=D58-LOC //S2 EXEC PGM=B,ACCT=D04+0
DD DSNAME	Hyphens (-)	DSNAME=A-B-C
	Periods to indicate a qualified data set name	DSNAME=A.B.C
	Double ampersands to identify a temporary data set name	DSNAME=&&TEMPDS
	Parentheses to enclose the member name of a partitioned data set, the area name of an indexed sequential data set, or the generation number of a generation data set	DSNAME=PDS1(MEMA) DSNAME=ISDS(PRIME) DSNAME=GDS(+1)
	Plus (+) or minus (-) sign to identify a generation of a generation data group	DSNAME=GDS(-2)
DD VOLUME=SER	Hyphens (-)	VOLUME=SER=PUB-RD
DD UNIT device-type	Hyphens (-)	UNIT=3330-1

Figure 4-4. Special Characters that Do Not Require Enclosing Apostrophes

Backward References

Many parameters in job control statements can use a backward reference to fill in information. A backward reference is a reference to an earlier statement in the job or in a cataloged or in-stream procedure called by a job step. A backward reference is in the form:

***.name** or ***.ddname** where name or ddname is the name field of the referenced statement.

***.stepname.name** or ***.stepname.ddname** where the referenced statement, name or ddname, is in an earlier step, stepname, in the same job.

***.stepname.procstepname.name** or ***.stepname.procstepname.ddname** where this job step or an earlier job step, stepname, calls a procedure; the procedure contains procedure step, procstepname, which contains the referenced statement, name or ddname.

The backward reference lets you copy previously coded information or refer to an earlier statement. The following parameters can make backward references:

- DD CNTL refers to earlier CNTL statement
- DD DCB refers to earlier DD statement to copy its DCB parameter
- DD DSNAME refers to earlier DD statement to copy its DSNAME parameter, whether or not the data set is a partitioned data set, and whether or not the data set is a temporary data set
- DD OUTPUT refers to earlier OUTPUT JCL statement
- DD REFDD refers to earlier DD statement to copy its data set attributes
- DD VOLUME=REF refers to earlier DD statement to use the same volume(s). The LABEL label type subparameter is also copied from the referenced DD statement.
- EXEC PGM refers to an earlier DD statement that defines the program to be executed as a member of a partitioned data set

The following statements cannot be referenced:

- DD * statement in DCB, DSNAME, or VOLUME parameter
- DD DATA statement in DCB, DSNAME, or VOLUME parameter
- DD DUMMY statement in VOLUME or UNIT parameter. The referring DD statement acquires a dummy status.
- DD DYNAM statement
- DD statement containing FREE=CLOSE in VOLUME or UNIT parameters
- Sysout DD statement

Examples of Backward References:

```
//JOB1     JOB   ...
//STEPA    EXEC  ...
//DD1      DD    DSNAME=REPORT
           .
           .
//DD4      DD    DSNAME=*.DD1
```

The referring and referenced DD statements are in the same step.

```
//JOB2     JOB   ...
//STEP1    EXEC  ...
//DDA      DD    DSNAME=D58.POK.PUBS01
           .
           .
//STEP2    EXEC  ...
//DDB      DD    DSNAME=*.STEP1.DDA
```

The referring and referenced DD statements are in different steps in the same job.

Cataloged procedure PROC1 contains:

```
//PS1      EXEC  ...
           .
           .
//PSTEP1   EXEC  ...
//DS1      DD    DSNAME=DATA1
//PSTEP2   EXEC  ...
//DS2      DD    DSNAME=DATA2
           .
```

The job contains:

```
//JOB5     JOB   ...
//CALLER   EXEC  PROC=PROC1
           .
//REF1     DD    DSNAME=*.CALLER.PSTEP2.DS2
//NEXT     EXEC  ...
//REF2     DD    DSNAME=*.CALLER.PSTEP1.DS1
           .
```

DD statement REF1 in the calling step refers to DD statement DS2 in procedure step PSTEP2. DD statement REF2 in a step after the calling step refers to DD statement DS1 in procedure step PSTEP1. Note that the entire procedure is processed when the calling EXEC statement is processed; therefore, all DD statements in the procedure are earlier than all DD statements in the calling step.

Chapter 5. Cataloged and In-Stream Procedures

For jobs that you run frequently or types of jobs that use the same job control, prepare sets of job control statements, called procedures. Place a procedure in an input stream, where it is called an **in-stream procedure**, and execute it to test it. Then place, or catalog, it in the system procedure library, SYS1.PROCLIB (or an installation-defined procedure library), where it is called a **cataloged procedure**.

Procedure Statements: A cataloged procedure consists of EXEC, DD, OUTPUT JCL, CNTL, ENDCNTL, and JCL comment statements and, optionally, may begin with a PROC statement. An in-stream procedure consists of the same statements and must begin with a PROC statement and end with a PEND statement.

The maximum number of in-stream procedures you can code in any job is 15. Each of these 15 procedures can be called more than once.

Note: Do not place any other JCL statements or any JES2 or JES3 control statements in a procedure. Do not place an in-stream data set, which begins with DD * or DD DATA, in a procedure.

Using a Procedure: To execute a procedure, call it on an EXEC statement in an in-stream job. On the EXEC statement, specify the name of the procedure in a PROC parameter. The step uses the JCL statements in the procedure as if the JCL statements appeared in the input stream immediately following the EXEC statement. If necessary, you can modify the procedure for the current execution of the job step.

If the called procedure is an in-stream procedure, place the procedure before the EXEC statement that calls it.

If the called procedure is cataloged, the system retrieves it from SYS1.PROCLIB (or an installation-defined procedure library) unless one of the following is specified:

- In a JES2 system, a PROCLIB parameter on a JES2 /*JOBPARM statement.
- In a JES3 system, a library name in a PROC parameter on a JES3 //*MAIN statement.

Testing a Procedure: Before putting a procedure into the procedure library, you should test it. For testing, place a PROC statement before the procedure and a PEND statement after it and place it in an input stream. For the test, call this procedure with an EXEC statement that appears later in the same job. After testing the procedure, catalog it; then call it with an EXEC statement whenever you want to use it.

Cataloged and in-stream procedures are not checked for correct syntax until an EXEC statement that calls the procedure is checked for syntax. Therefore, a procedure can be tested only if an EXEC statement calls it.

Cataloging a Procedure: The library containing cataloged procedures is a partitioned data set. The system procedure library is SYS1.PROCLIB. The installation can have many more procedure libraries with different names. When a cataloged procedure is called, the calling step receives a copy of the procedure; therefore, a cataloged procedure can be used simultaneously by more than one job.

Use the IEBUPDTE utility program to add a procedure to a library or to modify permanently a procedure in a catalog. 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.

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

Modifying Procedures

For its current execution, you can modify an in-stream or cataloged procedure by:

- Overriding, nullifying, or adding EXEC statement parameters
- Overriding, nullifying, or adding parameters to DD or OUTPUT JCL statements
- Adding DD or OUTPUT JCL statements

Invalid parameters in a procedure cannot be corrected through modification. Before making modifications, the system scans the original procedure statements for errors and issues error messages.

Modifying EXEC Statement Parameters

All keyword parameters on the calling EXEC statement affect the execution of the procedure, as follows:

- **All procedure statements:** If a keyword parameter is to override the parameter or be added to every EXEC statement in the procedure, code the parameter in the usual way. For example, the ACCT parameter applies to all steps:

```
//STEP1  EXEC  PROC=RPT,ACCT=5670
```

  Note: A PARM parameter without a procstepname qualifier applies only to the first procedure step. A TIME parameter without a procstepname qualifier applies to the entire procedure and nullifies any TIME parameters on procedure step EXEC statements.

  If the keyword parameter is to nullify the parameter on every EXEC statement in the procedure, code it without a value following the equal sign. For example, the ACCT parameter is nullified in all steps:

```
//STEP2  EXEC  PROC=RPT,ACCT=
```

- **Only one procedure statement:** If the keyword parameter is to override the parameter or be added to only one EXEC statement in the procedure, code **.procstepname** immediately following the keyword. The procstepname is the name field of the procedure EXEC statement containing the keyword parameter to be overridden. For example, the ACCT parameter applies to only step PSTEPWED:

```
//STEP1  EXEC  PROC=RPT,ACCT.PSTEPWED=5670
```

  If the keyword parameter is to nullify the parameter on only one EXEC statement in the procedure, code it with the procstepname. For example:

```
//STEP2  EXEC  PROC=RPT,ACCT.PSTEPTUE=
```

The override, nullifying, or addition applies only to the current execution of the job step; the procedure itself is not changed.

Rules for Coding Modifying EXEC Parameters:

- A PGM parameter cannot be modified.
- The calling EXEC statement can contain more than one change.
- Modifying parameters should appear in the following order:
  - Parameters without a procstepname qualifier.
  - All parameters modifying the first step, then the second step, then the third step, etc.
- The parameters for each step do not need to be coded in the same order as they appear on the procedure EXEC statement.
- An entire overriding parameter must be coded, even though only part of the overridden parameter is being changed.

Modifying OUTPUT JCL and DD Statements

OUTPUT JCL and DD statements following the calling EXEC statement (1) override, nullify, or add parameters to OUTPUT JCL and DD statements in the procedure or (2) are added to the procedure. These changes affect only the current execution of the job step; the procedure itself is not changed. When an OUTPUT JCL statement is modified, the sysout data set is processed according to the parameters as modified by the overriding statement.

In a procedure, to ensure that OUTPUT JCL and DD statements are overridden correctly by modifying statements, place the OUTPUT JCL statements before the DD statements in each step of the procedure.

Location in the JCL: Place modifying OUTPUT JCL and DD statements in the following order, after the EXEC statement that calls the procedure:

- For each procedure step in the invoked procedure:
  1. Overriding statements must appear in the same order as the statements that they are overriding.
  2. Added statements must appear after all overriding statements.

- For all procedure steps in the invoked procedure, place the modifying statements for each procedure step in the same order in which the procedure steps are specified.

Coding an Overriding OUTPUT JCL or DD Statement: To override, nullify, or add parameters to a procedure OUTPUT JCL or DD statement, code in the name field of the overriding OUTPUT JCL or DD statement the name of the procedure step containing the overridden statement, followed by a period, followed by the name of the procedure OUTPUT JCL statement or the ddname of the procedure DD statement.

```
//pstepname.name   OUTPUT parameters
//pstepname.ddname DD     parameters
```

Rules for Modifying OUTPUT JCL or DD Parameters:

- The overriding statement can contain more than one change.
- Modifying parameters can appear in any order.
- To nullify a parameter, do not code a value after the equal sign.
- If you are adding a parameter that is mutually exclusive with a parameter on the procedure statement, the procedure parameter is automatically nullified.
- An entire overriding parameter must be coded, even though only part of the overridden parameter is being changed.

Rules for Modifying DD Parameters:

- To nullify all parameters but the DCB parameter, code DUMMY on the overriding DD statement.
- Special rules apply when overriding a DCB parameter:
  - Code only the keyword subparameters to be changed; the other DCB subparameters remain unchanged.
  - If a positional subparameter is needed, code it, regardless of whether it appears in the overridden DCB parameter. If a positional subparameter is not needed or is to be nullified, omit it from the overriding DCB parameter.
  - To nullify the entire DCB parameter, nullify each subparameter appearing in the overridden DCB parameter.
- To nullify a DUMMY parameter on the procedure statement, code a DSNAME parameter on the overriding DD statement and assign a name other than NULLFILE.

Coding an Added OUTPUT JCL or DD Statement: To add OUTPUT JCL or DD statements to a procedure step, code in the name field of the added OUTPUT JCL or DD statement the name of the procedure step, followed by a period, followed by a name or ddname. The name must not appear on any procedure statement.

```
//pstepname.name   OUTPUT parameters
//pstepname.ddname DD     parameters
```

If you omit the procedure step name, the statement is added to the step named in the previous OUTPUT JCL or DD statement that named a step. If no previous statements named steps, the statement is added to the first step in the procedure.

Added OUTPUT JCL and DD statements can contain symbolic parameters. If the statement is being added to the last procedure step, any symbolic parameters it contains must appear elsewhere in the procedure.

In-stream Data for a Procedure: To supply a procedure step with data from the input stream, code a DD * or DD DATA statement in the calling step after the last overriding and added DD statement. The name field of this statement must contain the name of the procedure step, followed by a period, followed by a ddname. The ddname can be of your choosing or predefined in the procedure. If it is predefined, it appears in a DDNAME parameter on a procedure DD statement. For example:

```
//PROCSTP1.ANYNAME DD *
//PROCSTP2.PREDEFN DD DATA
```

Rules for Modifying DD Statements in Concatenated Data Sets:

- To override the first DD statement in a concatenation, code only one overriding DD statement.
- To override any following DD statements in the concatenation, code an overriding DD statement for each concatenated DD statement.
- The overriding DD statements must be in the same order as the concatenated DD statements.
- Code a ddname on only the first overriding DD statement. Leave the ddname field blank on the following statements.
- To leave a concatenated statement unchanged, code its corresponding overriding DD statement with a blank operand field.

Examples of Procedures

```
In the input stream:

//JOBA         JOB  ACCT23,'G. HILL'
//STEPA        EXEC PROC=REP
//PSTEP1.INDS DD   *
                .
                (data)
                .
/*

In SYS1.PROCLIB member REP:

//        PROC
//PSTEP1 EXEC PGM=WRIT22
//OUTDS  DD   SYSOUT=A
```

In this example, the EXEC statement STEPA calls the cataloged procedure named REP and supplies in-stream data. The procedure executes a program named WRIT22. The output from the program will appear in the sysout class A data set.

```
In the input stream:

//JOB1          JOB      ,'H.H. MORRILL'
//ADD1          OUTPUT   COPIES=2
//STEPA         EXEC     PROC=P
//PS1.OUTA      OUTPUT   CONTROL=DOUBLE,COPIES=5
//PS1.DSB       DD       OUTPUT=*.ADD1
//PS1.DSE       DD       *
                .
                (data)
                .
/*
//PS2.OUTB      OUTPUT   DEFAULT=YES,DEST=STL

In SYS1.PROCLIB member P:

//PS1           EXEC     PGM=R15
//OUTA          OUTPUT   CONTROL=PROGRAM
//DSA           DD       SYSOUT=C,OUTPUT=*.OUTA
//DSB           DD       SYSOUT=D,OUTPUT=*.OUTA
//PS2           EXEC     PGM=T48
//DSC           DD       SYSOUT=A
```

In this example, added statements are:

- ADD1, which is an OUTPUT JCL statement added at the job level.
- PS1.DSE, which is an in-stream data set added to procedure step PS1.
- PS2.OUTB, which is a default OUTPUT JCL statement added to procedure step PS2.

Overriding statements are:

- PS1.OUTA, which changes the CONTROL parameter and adds a COPIES parameter to OUTPUT statement OUTA in procedure step PS1.
- PS1.DSB, which changes the OUTPUT parameter on DD statement DSB in procedure step PS1.

```
//JOBB         JOB  ACCT23,'G. HILL'
//STEPB        EXEC PROC=WRIT35,COND.PSTEP3=(4,GT,PSTEP1),RD=R
//PSTEP1.DD1   DD   VOLUME=SER=,UNIT=SYSDA,DISP=(NEW,CATALG)
//PSTEP1.INDS  DD   *
                .
                .
                (data)
                .
/*
//PSTEP2.DD3   DD   DISP=(OLD,KEEP)
//PSTEP3.DD5   DD   DUMMY
//PSTEP3.DD6   DD   DSNAME=A.B.C
```

In SYS1.PROCLIB member WRIT35:

```
//        PROC
//PSTEP1  EXEC PGM=WT1,TIME=(,50)
//DD1     DD   DSNAME=DATA1,DISP=(NEW,DELETE),SPACE=(TRK,(10,2)),
//         UNIT=3330,VOL=SER=1095
//DD2     DD   DSNAME=&&WORK,UNIT=SYSDA,SPACE=(CYL,(10,1)),
//         DISP=(,PASS)
//PSTEP2  EXEC PGM=WT2,TIME=(,30)
//DD3     DD   DSNAME=*.PSTEP1.DD2,DISP=(OLD,DELETE)
//PSTEP3  EXEC PGM=UPDT,TIME=(,45),RD=RNC
//DD4     DD   SYSOUT=*
//DD5     DD   DSNAME=DATA3,UNIT=3340,DISP=OLD,
//         VOLUME=SER=335006
//DD6     DD   DSNAME=QOUT,UNIT=3400-5
//DD7     DD   SYSOUT=H
```

In this example, EXEC statement STEPB calls the cataloged procedure WRIT35. The COND parameter is added to the EXEC statement for PSTEP3. The RD parameter is added to the EXEC statements for PSTEP1 and PSTEP2, and overrides the RD parameter on the EXEC statement for PSTEP3.

In-stream DD statement PSTEP1.DD1 modifies DD statement DD1 in PSTEP1; it nullifies the VOLUME=SER parameter and overrides the UNIT and DISP parameters. Note that the parameters are not in the same order in the overriding and overridden statements.

In-stream DD statement PSTEP1.INDS is added to PSTEP1, supplying in-stream data to be read by program WT1.

In-stream DD statement PSTEP2.DD3 modifies DD statement DD3 in PSTEP2; it overrides the DISP parameter. Note that the entire parameter is coded, even though only the second subparameter is being changed.

In-stream DD statement PSTEP3.DD5 nullifies DD statement DD5 in PSTEP3. However, DD statement DD5 will be checked for correct syntax.

In-stream DD statement PSTEP3.DD6 modifies DD statement DD6 in PSTEP3; it overrides the DSNAME parameter.

Note that procedure DD statements DD2, DD4, and DD7 were not modified.

Symbolic Parameters

In order to be modified easily, the JCL statements in cataloged and in-stream procedures can contain **symbolic parameters**. A symbolic parameter can stand as a symbol for a parameter, a subparameter, or a value, that is, for any information in the parameter field of a procedure statement.

Purpose of Symbolic Parameters: Any parameter, subparameter, or value in a procedure that can vary each time the procedure is called is a good candidate to be coded as a symbolic parameter. For example, if a job step is charged to different account numbers each time the procedure is executed, code the ACCT parameter on the EXEC statement as one or more symbolic parameters. For example:

```
ACCT=&ALLNOS
ACCT=&FIRST&SECOND&THIRD
```

Coding Symbolic Parameters

Where: Code symbolic parameters on statements in a cataloged or in-stream procedure. You can also code symbolic parameters on DD statements being added to a procedure. However, on a DD statement being added to the last step of a procedure, do **not** use symbolic parameters that are not used previously in the procedure.

The records in an in-stream data set that are defined by the DD * or DATA parameter are not processed as JCL statements. Therefore, values are not substituted for any symbolic parameters in these records.

Syntax: A symbolic parameter consists of an ampersand (&) followed by a name, which is 1 through 7 alphanumeric and national ($, #, @) characters. The first character must be alphabetic or national. For example, DEST=&LOC on a DD or OUTPUT JCL statement.

Words Prohibited as Names in Symbolic Parameters: EXEC statement parameter and subparameter keywords cannot be used as the name in a symbolic parameter. For example, do not code &REGION=200K or REGION=&REGION; correctly code REGION=&SIZE.

In a procedure started by a START command from the operator console, do not use as symbolic parameters:

DD statement keywords
AFF
SEP
SPLIT
SUBALLOC

Symbolic Parameters Enclosed in Apostrophes: If a symbolic parameter is enclosed in apostrophes, correct substitution will occur only if the enclosed symbolic parameter is immediately preceded by a symbolic parameter that is not enclosed in apostrophes. For example, both A and B will be substituted correctly in:

```
//DD1  DD  &A'&B',DISP=OLD
```

Symbolic Parameter before Fixed Code: A period is required after a symbolic parameter when the following code does not vary and begins with:

- An alphanumeric or national character ($, #, @)
- A period

The system recognizes the period as a delimiter: the period does not appear after the symbolic parameter is assigned a value or nullified.

For example, if the first part of the data set name varies and the last does not, MONDATA, TUESDATA, etc., code:

```
DSNAME=&DAY.DATA
```

Code two consecutive periods (..) if a period should follow the symbolic parameter. For example, code &DEPT..POK when the desired value is D58.POK and DEPT=D58 is the value assignment.

Symbolic Parameter for Positional Parameter: When a symbolic parameter is a positional parameter followed by other parameters in the statement, the symbolic parameter should be followed by a **period instead of a comma**. For example:

```
//DS1    DD   &POSPARM.DSNAME=ATLAS,DISP=OLD
```

If &POSPARM is nullified, the statement appears as:

```
//DS1    DD   DSNAME=ATLAS,DISP=OLD
```

When assigning a value to &POSPARM, include the comma:

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

Symbolic Parameter after Fixed Code: When a symbolic parameter is placed after information that does not vary, the system recognizes the symbolic parameter when it encounters the ampersand. For example:

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

Two or More Symbolic Parameters: Code two or more symbolic parameters in succession without including a comma. For example:

```
PARM=&DECK&CODE
```

If the value should contain a comma, include a comma in the value assigned to the symbolic parameter.

Multiple Symbolic Parameters: The same symbolic parameter can appear more than once in a procedure, as long as the value assigned to the symbolic parameter is the same throughout the procedure. Therefore, &DEPT can appear several times in a procedure, if the department number is always to be the same.

Uniform Symbolic Names in an Installation: The names for symbolic parameters should be consistent in all the cataloged and in-stream procedures at an installation. For example, every time a department number is to be assigned to a symbolic parameter in any procedure in the installation, the symbolic parameter could be named &DEPT. Different procedures could contain ACCT=(43877,&DEPT) and DSNAME=LIBRARY.&DEPT..TALLY. The installation can tell all programmers the meaning of all the "standard" symbolic names.

Assigning Values to and Nullifying Symbolic Parameters

For each use of a procedure, all symbolic parameters must be assigned a value or be nullified. Any not assigned a value or nullified remain as coded in the JCL and may cause errors when the procedure is executed. To assign a value to or nullify a symbolic parameter, code the value on one or both of the following:

- The EXEC statement that calls the procedure. All symbolic parameters on the EXEC statement must appear in the procedure.
- The PROC statement that must begin an in-stream procedure and can begin a cataloged procedure.

The symbolic parameters can be coded in any order on the PROC and EXEC statements.

Multiple Values: If a value is assigned and/or nullified on both statements, the value on the EXEC statement overrides the value on the PROC statement.

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

Assigning Default Values to Symbolic Parameters: On the PROC statement, you should assign default values to all symbolic parameters in a procedure. The default will be used if a value is not assigned on the calling EXEC statement.

Assigning a Value to a Symbolic Parameter

To assign a value to symbolic parameter, code:

```
symbolic-parameter=value
```

Do not code the ampersand that identifies the symbolic parameter in the procedure.

For example, if the symbolic parameter UNIT=&NUMBER appears on a DD statement in the procedure, code NUMBER=value on the PROC or EXEC statement.

Length of Assigned Value: The value cannot be continued onto another statement.

The length of the value you assign, combined with the length of all parameters and delimiters in the parameter field of the single procedure statement containing the symbolic parameter, cannot exceed 120 characters. For example, if a procedure contains:

```
//DD1  DD  DSNAME=&A,DISP=(,PASS),UNIT=3350,
//       VOLUME=SER=25143,SPACE=(CYL,(10,10),,CONTIG)
```

The value assigned to &A must not be longer than 96 characters because the rest of the parameter field on that statement is 24 characters. Note that the length of the parameter field on the continuation statement is not considered.

Special Characters in the Assigned Value: If the value contains special characters, enclose the value in apostrophes. The enclosing apostrophes are not considered part of the value. If the special characters include apostrophes, code each apostrophe as two consecutive apostrophes. For example, LOC='O''HARE'.

Nullifying a Symbolic Parameter

To nullify a symbolic parameter, code:

```
symbolic-parameter=
```

Do not code the ampersand that identifies the symbolic parameter in the procedure. Do not code a value after the equal sign. Do not code literal blanks, that is, VALUE=' ', to nullify a symbolic parameter.

For example, if the symbolic parameter UNIT=&NUMBER appears on a DD statement in a procedure, code one of the following to nullify it:

```
//CALLER EXEC  PROC=ABC,NUMBER=,ACCT=DID58
//        PROC  NUMBER=,LOC=POK
```

If you nullify a symbolic parameter on the calling EXEC statement, it is nullified, even though the procedure's PROC statement contains a default value.

A symbolic parameter that is last on a statement that is being continued cannot be nullified; it must be assigned a value. An attempt to nullify such a parameter results in a JCL error.

Cautions about Leading and Trailing Commas: When a symbolic parameter is nullified, a delimiter, such as a leading or trailing comma, is not removed. In some cases, the remaining comma is needed; in others, it will be a syntax error.

When Symbolic Parameter is Positional

If a symbolic parameter is a positional parameter, a comma must remain to indicate its omission if another positional parameter follows. In this case, code a comma before and after the symbolic parameter; the needed commas will remain after nullification. For example, &NUMBER for the unit count:

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

When &NUMBER is nullified, the parameter correctly becomes:

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

When Symbolic Parameter is Not Positional

If a symbolic parameter is not a positional parameter, a comma must not remain if it is omitted. In this case, do not code a comma before the symbolic parameter; no commas will remain after nullification. For example, serial numbers in the VOLUME=SER parameter:

```
VOLUME=SER=(&FIRST&SECOND)
```

If either of the symbolic parameters is nullified, a leading or trailing comma will not remain. If you nullify &FIRST and assign 222222 for &SECOND or if you nullify &SECOND and assign 111111 for &FIRST, the parameter correctly becomes:

```
VOLUME=SER=(222222)
VOLUME=SER=(111111)
```

Code a comma when you assign a value to the symbolic parameter and a comma is needed. For example:

```
//CALLER  EXEC  PROC=ABC,FIRST=111111,SECOND=',222222'
```

Because the comma is a special character, enclose the value in apostrophes.

Examples of Symbolic Parameters

```
//JOBA      JOB  ...
//INSTREAM  PROC LOC=POK
//PSTEP     EXEC PGM=WRITER
//DSA       DD   SYSOUT=A,DEST=&LOC
//          PEND
//CALLER    EXEC PROC=INSTREAM,LOC=NYC
//
```

In this example of an in-stream procedure, the symbolic parameter &LOC is given a default value of POK on the PROC statement. Then it is given a current execution value of NYC on the calling EXEC statement.

```
//JOBB       JOB  ...
//INSTREAM   PROC LOC=POK,NUMBER=3350
//PSTEP      EXEC ...
//PIN        DD   DSNAME=REPORT,DISP=(OLD,KEEP),UNIT=&NUMBER
//POUT       DD   SYSOUT=A,DEST=&LOC
//           PEND
//CALLER     EXEC PROC=INSTREAM,NUMBER=,LOC=STL
//PSTEP.INDATA  DD   *
             .
             (data)

/*
```

This code nullifies the symbolic parameter &NUMBER. The calling EXEC statement assignment of STL for symbolic parameter &LOC overrides the PROC statement assignment of POK.

This example illustrates execution of an in-stream procedure to test symbolic parameters before placing the procedure in a procedure library. The in-stream procedure named TESTPROC is:

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

To execute this in-stream procedure and override &A with IEFBR14, &B with BAKER, and &E with (NEW, KEEP) but leave the other parameters the same, call the in-stream procedure with:

```
//CALLER1 EXEC PROC=TESTPROC,A=IEFBR14,B=BAKER,E='(NEW,KEEP)'
```

After the symbolic substitution, the statements are:

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

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

```
//CALLER2 EXEC PROC=TESTPROC,A=IEFBR14,B=,C=3350,D=,E=
```

After the symbolic substitution, the statements are:

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

Chapter 6. Job Control Statements on the Output Listing

Use the JOB statement MSGLEVEL parameter to request that job control statements be printed in the job log output listing. Code MSGLEVEL=(1,1) to receive the maximum amount of information, in the following order:

- JES messages and job statistics.
- All job control statements in the input stream and procedures.
- Messages about job control statements.
- JES and operator messages about the job's processing: allocation of devices and volumes, execution and termination of job steps and the job, and disposition of data sets.

Statements in Listing: To identify the source and type of each statement, the system prints certain characters in columns 1 and 2 or 1, 2, and 3 of the listing. These identifying characters are explained in Figure 6-1 on page 6-2. The listing shows all procedure statements as they appear in the cataloged procedure; the listing does not show parameter substitutions and overrides on the statement itself.

Symbolic Parameters: The job log listing shows the symbolic parameters in procedure statements. The values assigned to the parameters are given in IEF653I messages. These messages appear immediately after each statement that contains symbolic parameters.

EXEC Overriding Parameters: A procedure EXEC statement appears in the job log listing exactly as it appears in the procedure. Overridden parameters must be shown by the program being executed:

- For the EXEC statement that executes the assembler program, the **Diagnostic Cross Reference and Assembler Summary** produced by the assembler program shows the overriding parameters.
- For the EXEC statement that executes the linkage editor, the linkage editor listing shows the overriding parameters.

Columns 1, 2, and 3	Source and Type of Statement
Job Control Statements in the Input Stream	
//	JCL statement
//*	Job control statement that is not a JCL comment statement but one that the system considers to contain only comments
***	JES2 statement
***	JES3 statement
***	JCL comment statement
Cataloged Procedure Statements	
XX	DD statement that was not overridden and all other JCL statements, except the JCL comment statement. Each statement appears in the listing exactly as it appears in the procedure.
X/	DD statement that was overridden (preceded by the overriding DD statement)
XX*	Job control statement that is not a JCL comment statement but one that the system considers to contain only comments
***	JCL comment statement
In-Stream Procedure Statements	
+ +	DD statement that was not overridden and all other JCL statements, except the JCL comment statement. Each statement appears in the listing exactly as it appears in the procedure.
+/	DD statement that was overridden (preceded by the overriding DD statement)
+ +*	Job control statement that is not a JCL comment statement but one that the system considers to contain only comments
***	JCL comment statement

Figure 6-1. Identification of Statements in Job Log

Chapter 7. JCL Command Statement

Purpose: Use the JCL command statement to enter an MVS operator command through the input stream.

The JCL command statement is supported only on JES2 systems.

Note: To enter a JES2 command, use the JES2 command statement described on page 21-2. To enter a JES3 command, use the JES3 command statement described on page 22-3.

The system usually executes an in-stream command as soon as it is read. Therefore, the command will **not** be synchronized with the execution of any job or step in the input stream. To synchronize a command with the job processing, tell the operator the commands you want and when they should be issued, and let the operator enter them from the console.

The system processes each command according to installation options for both the input device from which the job was read, and the job class.

References: For more information on MVS commands and for descriptions of their parameters, see *System Commands.*

Syntax:

```
//  command  [parameter] [comments]
```

The command statement consists of the characters // in columns 1 and 2 and three fields: operation (command), parameter, and comments.

Do not continue a command statement.

JCL Command Statement

Operation Field

The operation field contains the MVS operator command and is coded as follows:

- Precede and follow the command with one or more blanks. It can begin in any column.
- Code the command or a valid abbreviation for the command. The following MVS operator commands can be entered through the input stream.

CANCEL	MONITOR	SEND	STOP
CHNGDUMP	MOUNT	SET	STOPMN
DISPLAY	PAGEADD	SETDMN	UNLOAD
HOLD	RELEASE	SLIP	VARY
LOG	REPLY	START	WRITELOG
MODIFY	RESET		

Parameter Field

Code any required parameters. When more than one parameter is coded, separate them with commas.

Comments Field

The comments field follows the parameter field after at least one intervening blank.

Location in the JCL

A command statement can appear anywhere after a JOB statement and before the end of the job. If a command statement appears between jobs, it is ignored. A command statement should not be placed before the first JOB statement in an input stream.

If a command statement contains errors, it is not executed. If the erroneous statement is between two jobs in the input stream, the system does not issue a message to indicate that the command is not executed.

Example of the Command Statement

```
//   DISPLAY TS,LIST
```

In response to this command statement, the system displays the number and userid of all active time-sharing users of the system.

Chapter 8. Comment Statement

Purpose: Use the comment statement to enter a comment on the output listing. The comment statement is used primarily to document a job and its resource requirements.

Syntax:

```
//*comments
```

The comment statement consists of the characters //* in columns 1, 2, and 3 and one field: comments.

Code the comments in columns 4 through 80. The comments field does not need to be preceded or followed by blanks. (In a JES3 system, do not use a JES3 keyword as the first word in column 4 of the comment field, or the comment might be taken for a JES3 statement.)

Do not continue a comment statement using continuation conventions. Instead, code additional comment statements.

Location in the JCL

Place a comment statement anywhere after the JOB statement. You can place a comment statement between continuations of JCL statements.

Listing of Comments Statements

Use the MSGLEVEL parameter on the JOB statement to request that the job log output listing contain all the JCL statements for your job. On this listing, comment statements have *** in columns 1, 2, and 3.

Examples of the Comment Statement

```
//* THE COMMENT STATEMENT CANNOT BE CONTINUED,
//* BUT IF YOU HAVE A LOT TO SAY, YOU CAN FOLLOW A
//* COMMENT STATEMENT WITH MORE COMMENT
//* STATEMENTS.
```

Chapter 9. CNTL Statement

Purpose: Use the CNTL statement to mark the beginning of program control statements in the input stream. Program control statements specify control information for a subsystem. The program control statements are ended by an ENDCNTL statement and are called a CNTL/ENDCNTL group.

The DD statement that defines a data set to be processed by a subsystem must refer to the CNTL statement in order for the subsystem to use the program control statements in processing the data set.

References: The program control statements are documented in the publications for the subsystems. For example, see *IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide* for information on program control statements for the Print Services Facility (PSF).

Syntax:

```
//label CNTL  [ *  comments]
```

The CNTL statement consists of the characters // in columns 1 and 2 and four fields: label, operation (CNTL), parameter (*), and comments. The * parameter is required only when comments follow.

Label Field

Code a label on every CNTL statement, as follows:

- Each label must be unique within the job.
- The label must begin in column 3.
- The label is 1 through 8 alphanumeric or national ($, #, @) characters.
- The first character must be alphabetic or national ($, #, @)
- The label must be followed by at least one blank.

Operation Field

The operation field consists of the characters CNTL and must be preceded and followed by at least one blank. It can begin in any column.

CNTL

Parameter Field

The parameter field contains only an asterisk. When present, the asterisk must be preceded and followed by at least one blank. The asterisk is required only when the statement contains comments.

Comments Field

The comments field follows the asterisk after at least one intervening blank.

Location in the JCL

A CNTL statement must appear before the DD statement that refers to it. The CNTL and its referencing DD statement must be in the same job step or in the same cataloged or in-stream procedure step. A CNTL statement can be in a procedure and the referencing DD statement can be in the calling job step, but not vice versa.

Program Control Statements

Program control statements supply control information for a subsystem. A subsystem can require one or more program control statements. The one or more statements must be immediately preceded by a CNTL statement and immediately followed by an ENDCNTL statement.

Do not code JCL statements within a program control group.

Program Control Statements in Procedures

You can code symbolic parameters on program control statements in a cataloged or in-stream procedure.

You can override parameters on program control statements in a procedure. Follow the rules used for overriding DD statement parameters in a procedure. For more information, see "Modifying OUTPUT JCL and DD Statements" on page 5-3.

Example of the CNTL Statement

```
//STEP1     EXEC  PGM=PRINT
//ALPHA     CNTL  *    PROGRAM CONTROL STATEMENT FOLLOWS
//PRGCNTL   PRINTDEV BUFNO=20,PIMSG=YES,DATACK=BLOCK
//OMEGA     ENDCNTL
//AGAR      DD    UNIT=3800-3,CNTL=*.ALPHA
```

The PSF subsystem uses the BUFNO, PIMSG, and DATACK options of the PRINTDEV control statement to print the data set for DD statement AGAR on a 3800 model 3.

Chapter 10. DD Statement

Purpose: Use the DD (data definition) statement to describe a data set and to specify the input and output resources needed for the data set.

The parameters you can specify for data set definition are arranged alphabetically in the following pages.

References: For information about the JES initialization parameters that provide installation defaults, see *SPL: JES2 Initialization and Tuning* and *SPL: JES3 Initialization and Tuning*.

Syntax:

```
//[ddname              ] DD [positional-parameter][,keyword-parameter]...
 [procstepname.ddname] [comments]

//[ddname              ] DD
 [procstepname.ddname]
```

- The DD statement consists of the characters // in columns 1 and 2 and four fields: name, operation (DD), parameter, and comments. Do not code comments if the parameter field is blank.
- A DD statement is required for each data set.
- The maximum number of DD statements per job step are:
  - 3273, in a JES2 system. Note that the limit of 3273 is based on the number of single DD statements allowed for a TIOT (task input output table) control block size of 32K. This limit can be different depending on the installation-defined TIOT size.
  - Determined by the installation, in a JES3 system.

Name Field

When specified, code a ddname as follows:

- Each ddname should be unique within the job step. If duplicate ddnames appear in a job step, processing is as follows:
  - **In a JES2 system**: The system performs device and space allocation and disposition processing for both DD statements; however, it directs all references to the first DD statement in the step.

  - **In a JES3 system**: If both DD statements request JES3 or jointly-managed devices, the system cancels the job during JES3 interpretation. If only one or neither DD statement requests a JES3 or jointly-managed device, the system performs device and space allocation processing for both DD statements; however, it directs all references to the first DD statement in the step.

- The ddname must begin in column 3.
- The ddname is 1 through 8 alphanumeric or national ($, #, @) characters.
- The first character must be alphabetic or national ($, #, @).
- The ddname must be followed by at least one blank.

Omitting the ddname: Do not code a ddname in two cases:

- The DD statement defines a data set that is concatenated to the data set of the preceding DD statement.
- The DD statement is the second or third consecutive DD statement for an indexed sequential data set.

ddname when Overriding or Adding to Procedures: On a DD statement that overrides a procedure DD statement, code in the name field the name of the procedure step containing the overridden DD statement, followed by a period, followed by the ddname of the procedure DD statement to be overridden. On a DD statement that is to be added to a procedure, code in the name field the name of the procedure step, followed by a period, followed by a ddname of your choosing. For example:

```
//PROCSTP1.DDA DD parameters
```

Special ddnames: Use the following special ddnames only when you want to use the facilities these names represent to the system. These facilities are explained in Chapter 11, "Special DD Statements."

JOBCAT	SYSCHK
JOBLIB	SYSCKEOV
STEPCAT	SYSIN
STEPLIB	SYSMDUMP
SYSABEND	SYSUDUMP

The following ddnames have special meaning to JES3; do not use them on a DD statement in a JES3 system.

JCBIN	JESJCL	JS3CATLG
JCBLOCK	JESMSG	J3JBINFO
JCBTAB	JOURNAL	J3SCINFO
JCLIN	JST	
JESInnnn	SYSMSG	

Operation Field

The operation field consists of the characters DD and must be preceded and followed by at least one blank. It can begin in any column.

Parameter Field

A DD statement has two kinds of parameters: positional and keyword. All parameters are optional. However, leave the parameter field blank only on a DD statement that overrides a DD statement for a concatenated data set in a cataloged or in-stream procedure.

Positional Parameters: A DD statement can contain **one** positional parameter. If coded, this positional parameter must precede all keyword parameters.

POSITIONAL PARAMETERS	VALUES	PURPOSE
[*|DATA] See page 10-13 or 10-39	*: for data sets containing no JCL DATA: for data sets containing JCL	Begins an in-stream data set.
DUMMY See page 10-86		Specifies no space allocation, no disposition processing, and, for BSAM and QSAM, no I/O.
DYNAM See page 10-89		(Parameter is supported to provide compatibility with previous systems.)

Keyword Parameters: A DD statement can contain the following keyword parameters. You can code any of the keyword parameters in any order in the parameter field after a positional parameter, if coded.

Do not use DD statement keywords as symbolic parameters in procedures to be started by a START command from the operator console.

KEYWORD PARAMETERS	VALUES	PURPOSE
ACCODE=access-code See page 10-16	access-code: 1 - 8 characters, first must be upper case A - Z	Specifies or changes an accessibility code for an ISO/ANSI/FIPS Version 3 tape output data set.

KEYWORD PARAMETERS	VALUES	PURPOSE
AMP = (subparameter) AMP = ('subparameter[,subparameter]...') subparameters: AMORG BUFND = number BUFNI = number BUFSP = bytes CROPS = { RCK / NCK / NRE / NRC } OPTCD = { I / L / IL } RECFM = { F / FB / V / VB } STRNO = number SYNAD = modulename TRACE See page 10-18	see *VSAM Administration Guide*	Completes information in an access method control block (ACB) for a VSAM data set.
With SMS only: AVGREC = { U / K / M } See page 10-24	U: space specified in records K: space specified in thousands of records M: space specified in millions of records	Specifies a record request and the quantity of primary and secondary space specified on the SPACE parameter.
BURST = { YES / Y / NO / N } See page 10-26	YES or Y: burster-trimmer-stacker NO or N: continuous forms stacker	Directs output to a stacker on a 3800 Printing Subsystem.
CHARS = { table-name / (table-name[,table-name]...) / DUMP / DUMP[,table-name]... } See page 10-28	1 - 4 table-name subparameters: 1 - 4 alphanumeric or $, #, @ characters DUMP: 204-character print lines on 3800	Names character-arrangement tables for printing on a 3800 Printing Subsystem. Requests a high-density dump on a SYSABEND or SYSUDUMP DD statement.
CHKPT = EOV See page 10-31		Requests a checkpoint at each end-of-volume, except the last.
CNTL = { *.label / *.stepname.label / *.stepname.procstepname.label } See page 10-33	label: names CNTL statement stepname: CNTL in named step procstepname: step in named procedure	Causes the system to execute statements following an earlier CNTL statement.
COPIES = { nnn / (nnn,(group-value[,group-value]...)) / (,group-value[,group-value]...)) } See page 10-35	nnn (JES2): 1 - 255 nnn (JES3): 1 - 254 1 - 8 group-values (JES2): 1 - 255 1 - 8 group values (JES3): 1 - 254	Specifies number of copies printed. For a 3800 Printing Subsystem, can instead specify number of copies of each page printed before the next page is printed.

KEYWORD PARAMETERS	VALUES	PURPOSE
With SMS only: DATACLAS=data-class-name See page 10-42	data-class-name: installation-defined name of a data class	Specifies the data class for a new data set.
DCB=(subparameter[,subparameter]...) DCB= ({dsname *.ddname *.stepname.ddname *.stepname.procstepname.ddname} [,subparameter]...) See page 10-44	subparameter: see tables in DCB parameter description dsname: copy DCB information from named cataloged data set *.ddname: copy DCB parameter from named earlier DD statement stepname: DD in named step procstepname: step in named procedure	Completes information in data control block (DCB).
DDNAME=ddname See page 10-59	ddname: names later DD statement	Postpones defining the data set until later in same step: on a DD statement in the calling step or in a procedure called by the step.
DEST=destination destination (JES2): LOCAL name Nnnnn NnnRmmmm to NnnnnRmm Rnnnn or RMnnnn or RMTnnnn Unnnn (node,userid) destination (JES3): ANYLOCAL device-name device-number group-name nodename (node,userid) See page 10-63	LOCAL or ANYLOCAL: local device name: named local or remote device Nnnnn: node (1 - 1000) NnRm: node (1 - 1000) and remote work station (1 - 9999); 6 digits maximum for n and m combined Rnnnn or RMnnnn or RMTnnnn: remote terminal (1 - 9999) Unnnn: local terminal (1 - 9999) (node,userid): node (1 - 8 alphanumeric or $, #, @ characters) and TSO userid (1 - 7 alphanumeric or $, #, @ characters) or VM userid (1 - 8 alphanumeric or $, #, @ characters) device-number: 3-number address device-name: local device (1 - 8 alphanumeric or $, #, @ characters) group-name: 1 or more local devices or remote stations (1 - 8 alphanumeric or $, #, @ characters) nodename: node (1 - 8 alphanumeric or $, #, @ characters)	Sends a sysout data set to the specified destination.
DISP=status DISP=([status][,normal-termination-disp] [,abnormal-termination-disp]) See page 10-67	status: NEW, OLD, SHR (for shared), MOD (for data set to be modified) normal-termination-disp: DELETE, KEEP, PASS, CATLG, or UNCATLG abnormal-termination-disp: DELETE, KEEP, CATLG, or UNCATLG	Describes the status of the data set and tells the system to do the following with the data set after normal or abnormal termination of the step or job: delete or keep it on its volume(s), pass it to a later step, or add it to or remove it from the catalog.
DLM=delimiter See page 10-77	delimiter: 2 characters	Terminates an in-stream data set.

DD

KEYWORD PARAMETERS	VALUES	PURPOSE
DSID = { id / (id,[V]) } See page 10-79	id: 1 - 8 characters V: label was verified (only on a SYSIN DD statement)	Identifies a data set on a diskette of a 3540 Diskette Input/Output Unit.
{ DSNAME / DSN } = { dsname / dsname(member-name) / dsname(generation-number) / dsname(area-name) / &&dsname / &&dsname(member-name) / &&dsname(area-name) / *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname / NULLFILE } See page 10-81	unqualified dsname: 1 - 8 alphanumeric or $, #, @ characters, -, +0 qualified dsname: multiple names joined by periods member-name: member in PDS generation-number: 0 or signed integer area-name: INDEX, PRIME, or OVFLOW area in indexed sequential data set *.ddname: copy dsname from earlier DD stepname: DD in named step procstepname: step in named procedure NULLFILE: dummy data set	Names the data set.
EXPDT = { yyddd / yyyy/ddd } See page 10-90	yyddd: expiration date (yy: 2-digit year, ddd: day 001-366) yyyy/ddd: expiration date (yyyy: 4-digit year, ddd: day 001-366)	Specifies an expiration date for the data set.
FCB = { fcb-name / (fcb-name [,ALIGN] [,VERIFY]) } See page 10-92	fcb-name: 1 - 4 alphanumeric or $, #, @ characters ALIGN: operator check forms alignment VERIFY: operator verify FCB image	Specifies FCB image, carriage control tape for 1403 Printer, or data-protection image for 3525 Card Punch.
FLASH = { overlay-name / (overlay-name[,count]) / NONE } See page 10-96	overlay-name: forms overlay frame (1 - 4 alphanumeric or $, #, @ characters) count: copies with overlay (0 - 255) NONE: suppresses flashing	For printing on a 3800 Printing Subsystem, indicates that the data set is to be printed with the named forms overlay and can specify how many copies are to be flashed.
FREE = { END / CLOSE } See page 10-99	END: deallocate at end of step CLOSE: deallocate when data set is closed	Specifies when to deallocate the resources for this data set.
KEYLEN = bytes See page 10-104	bytes: number of bytes (1-255 for key-sequenced (KS), 0-255 for sequential (PS) or partitioned (PO))	Specifies the length of the keys in the data set.
With SMS only: KEYOFF = offset-to-key See page 10-106	offset-to-key: position of key (0 to difference of LRECL and KEYLEN minus 1)	Specifies the offset of the first byte of the record key.
HOLD = { YES / Y / NO / N } See page 10-102	YES or Y: holds this sysout data set NO or N: allows normal processing for this sysout data set's output class	Tells the system to hold this sysout data set until released by the operator.

KEYWORD PARAMETERS	VALUES	PURPOSE
LABEL = ([data-set-seq-no][,label-type] [,PASSWORD / ,NOPWREAD / ,] [,IN / ,OUT / ,] [,RETPD = nnnn / ,EXPDT = yyddd / yyyy/ddd]) See page 10-108	data-set-seq-no: data set position on tape volume (1 - 4 decimal digits) label-type: SL: IBM standard labels SUL: IBM standard and user labels AL: ISO/ANSI Version 1 and ISO/ANSI/FIPS Version 3 labels AUL: user labels and ISO/ANSI Version 1 and ISO/ANSI/FIPS Version 3 labels NSL: nonstandard labels NL: no labels BLP: bypass label processing LTM: leading tapemark PASSWORD: password required to access data set NOPWREAD: password required to change or delete data set IN: only read BSAM data set opened for INOUT or BDAM data set opened for UPDAT OUT: only write to BSAM data set opened for OUTIN or OUTINX RETPD = nnnn: retention period (nnnn: 1 - 4 decimal digits) EXPDT = yyddd: expiration date (yy: 2-digit year, ddd: day 001 - 366) EXPDT = yyyy/ddd: expiration date (yyyy: 4-digit year, ddd: day 001 - 366)	Specifies information about a data set's label, password, opening, expiration date, and, for a tape data set, relative position on the volume.
With SMS only: LIKE = data-set-name See page 10-115	data-set-name: dsname of model data set	Specifies the attributes of a new data set.
LRECL = bytes See page 10-117	bytes: length in bytes (1-32760 for PS or PO, 1-32761 for KS, ES, or RR)	Specifies the length of the records in the data set.
With SMS only: MGMTCLAS = data-class-name See page 10-119	data-class-name: installation-defined name of a data class	Specifies the management class for a new data set.
MODIFY = { module-name / (module-name[,trc]) } See page 10-121	module-name: 1 - 4 alphanumeric or $, #, @ characters trc: table-name in CHARS parameter (0 for first, 1 for second, 2 for third, and 3 for fourth table-name)	Specifies a copy-modification module in SYS1.IMAGELIB to be used by JES to print the data set on a 3800 Printing Subsystem.
MSVGP = { id / (id[,ddname]) / SYSGROUP } See page 10-123	id: group of mass storage volumes (1 - 8 alphanumeric or $, #, @ characters) ddname: allocates this data set to volume(s) other than those for DD statement ddname SYSGROUP: default group of mass storage volumes	Places the data set on a group of mass storage volumes on an MSS device.
OUTLIM = number See page 10-126	number: 1 - 16777215 logical records maximum	Limits the logical records in this sysout data set.

KEYWORD PARAMETERS	VALUES	PURPOSE
OUTPUT = { reference / (reference[,reference]...) } reference: *.name *.stepname.name *.stepname.procstepname.name See page 10-128	name: names earlier OUTPUT JCL statement stepname: OUTPUT JCL in named step procstepname: step in named procedure	Associates this sysout data set with one or more OUTPUT JCL statements.
PROTECT = YES See page 10-132		Requests that RACF create a discrete profile to protect a data set on direct access or a tape volume.
QNAME = procname[.tcamname] See page 10-135	procname: names a TPROCESS macro that defines a destination queue for the messages tcamname: names a TCAM job or started task to process the messages	Indicates that this data set contains TCAM messages.
RECFM = { F / FB / FBS / FS / V / VB / VBS / VS / U } [A / M] See page 10-136	Record format is: F: fixed length B: blocked S: spanned V: variable length U: undefined length Control characters are: A: ISO/ANSI code M: machine code	Specifies the format and characteristics of the records in a data set.
With SMS only: RECORG = { KS / ES / RR / LS } See page 10-141	Organization of records: KS: key-sequenced ES: entry-sequenced RR: relative record LS: linear space	Specifies the organization of the records in a VSAM data set.
With SMS only: REFDD = { *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname } See page 10-143	Referenced DD statement: ddname: unqualified name stepname: qualified by step name procstepname: step in procedure	Specifies the attributes of a new data set by referring to a previous DD statement.
RETPD = nnnn See page 10-145	nnnn: number of days (0-9999)	Specifies the retention period for a new data set.

KEYWORD PARAMETERS	VALUES	PURPOSE
With SMS only: SECMODEL=(profile-name[,GENERIC]) See page 10-147	profile-name: name of model profile GENERIC: model is generic profile	Specifies a RACF profile to be used for a new data set.
For system assignment of space: SPACE= (TRK, / CYL, / blklgth, / reclgth,) (primary-qty [,second-qty [,]] [,directory / ,index]) [,RLSE [,]] [,CONTIG / ,MXIG / ,ALX [,]] [,ROUND] To request specific tracks: SPACE= (ABSTR,(primary-qty,address [,directory / ,index])) To request directory blocks (with SMS only): SPACE=(,(,,directory)) See page 10-149	TRK: allocation in tracks CYL: allocation in cylinders blklgth: allocation in average blocks, 1 - 65535 reclgth: allocation in average records (SMS) primary-qty: number of tracks, cylinders, or blocks to be allocated second-qty: additional tracks or cylinders to be allocated, if more are needed directory: number of 256-byte records for PDS directory index: tracks or cylinders for index of indexed sequential data set RLSE: release unused space when data set is closed CONTIG: contiguous primary allocation MXIG: allocation in largest available space (not supported for indexed sequential data sets) ALX: allocation of up to 5 separate contiguous primary quantities ROUND: allocation by block length rounded to integral cylinders ABSTR: allocation at the specified address address: track number of first track to be allocated	Requests space for a new data set on direct access storage.
With SMS only: STORCLAS=storage-class-name See page 10-156	storage-class-name: installation-defined name of a storage class	Specifies the storage class for a new data set.
SUBSYS=(subsystem-name [,subsystem-parameter]...) See page 10-158	subsystem-name: identifies the subsystem subsystem-parameter: specifies information for the subsystem	Requests a subsystem to process this data set.
SYSOUT=class SYSOUT= ([class] [,writer-name / ,INTRDR [,]] [,form-name / ,code-name]) SYSOUT=* SYSOUT=(,) See page 10-161	class: A - Z, 0 - 9 writer-name: 1 - 8 alphanumeric or $, #, @ characters form-name: 1 - 4 alphanumeric or $, #, @ characters code-name: 1 - 4 alphanumeric or $, #, @ characters (JES2 only) *: same output class as MSGCLASS parameter on JOB statement	Defines this data set as a sysout data set and (1) assigns it to an output class, (2) requests external writer to process it, (3) identifies print or punch forms, and (4) refers to the code-name of a JES2 /*OUTPUT statement.

KEYWORD PARAMETERS	VALUES	PURPOSE
TERM=TS See page 10-168		In a foreground job, indicates that this data set is coming from or going to a TSO userid. In a batch job, indicates that this DD statement begins an in-stream data set.
UCS= {character-set-code (character-set-code [,FOLD] [,VERIFY])} See page 10-170	character-set-code: 1 - 4 alphanumeric or $, #, @ characters FOLD: operator load chain or train in fold mode VERIFY: operator verify UCS image	Specifies universal character set, print train, or character-arrangement table for a 3800 Printing Subsystem.
UNIT=({device-number / device-type / group-name}[,unit-count / ,P][,DEFER]) UNIT=AFF=ddname See page 10-173	device-number: 3-digit hexadecimal number device-type: machine type and model group-name: 1 - 8 alphanumeric or $, #, @ characters unit-count: 1 - 59 P: allocate same number of devices as volumes for parallel mount DEFER: defers mounting until open AFF=ddname: requests allocation of same devices as for DD statement ddname	Requests allocation to a specific device, a type or group of devices, or the same device(s) as another data set. Also can specify how many devices and deferred mounting.
{VOLUME / VOL}= ([PRIVATE][,RETAIN] [,volume-seq-no] [,volume-count] [,] {SER=(serial-number[,serial-number]...) / REF=dsname / REF=*.ddname / REF=*.stepname.ddname / REF=*.stepname.procstepname.ddname / REF=*.procstepname.ddname}) See page 10-178	PRIVATE: requests a private volume RETAIN: requests private tape volume remain mounted and unwound or requests public tape volume be retained at device volume-seq-no: begins processing with volume 1 - 255 of existing multivolume data set volume-count: maximum volumes for output data set (1 - 255) serial-number subparameters (1 - 255): volume serial numbers (1 - 6 alphanumeric, $, #, @, or special characters) REF: copy volume serial numbers from another data set or earlier DD statement, or copy storage class for SMS-managed data sets dsname: from cataloged or passed data set ddname: from named earlier DD statement stepname: DD in named step procstepname: step in named procedure	Identifies the volume(s) on which a data set resides or will reside.

Comments Field

The comments field follows the parameter field after at least one intervening blank. If you do not code any parameters on a DD statement, do not code any comments.

Location in the JCL

Most DD statements define data sets to be used in a job step, in a cataloged procedure step, or in an in-stream procedure step; these appear after the EXEC statement for the step. Some DD statements define data sets for the job, for example, the JOBLIB DD statement; these appear after the JOB statement and before the first EXEC statement.

When Overriding or Adding to Procedures: Place DD statements that override, nullify, or add parameters immediately following the EXEC statement that calls the procedure. Place overriding and nullifying DD statements first, followed by all added DD statements. Last in the calling step are any DD * or DD DATA statements with their in-stream data.

To override more than one DD statement in a procedure, place the overriding DD statements in the same order as the overridden DD statements in the procedure.

Concatenating Data Sets

You can logically connect or **concatenate** sequential or partitioned input data sets for the duration of a job step. Each of the concatenated data sets can reside on a different volume. For details on concatenating data sets, see *Data Administration Guide*.

Coding a Concatenation: To concatenate data sets, omit the ddnames from all the DD statements except the first in the sequence. The data sets are processed in the same sequence as the DD statements defining them.

Devices for Concatenated Data Sets: Concatenated data sets can reside on different devices and different types of devices. (This may require internal DCB modifications, see *Data Administration Guide*.) Also, do not concatenate data sets on direct access devices with Rotational Position Sensing (RPS) and data sets on non-RPS devices unless your application can handle this situation.

Block Sizes for Concatenated Data Sets: Concatenated data sets can have different block sizes as long as the data set with the largest block size appears first in the concatenation. (Note that you can state a value equal to the largest block size for BLKSIZE on the first DD statement, regardless of what the actual block size of this data set is. Also, certain data sets can be concatenated in any order of block size; these are (1) partitioned data sets, and (2) DASD-resident sequential data sets that are accessed by QSAM and use system-created buffers.)

The system determines an optimum block size for you if you do not specify a block size.

Logical Record Lengths for Concatenated Data Sets: Concatenated data sets can have different logical record lengths as long as the data set with the largest logical record length appears first in the concatenation. (Note that you can state a value equal to the largest logical record length for LRECL on the first DD statement, regardless of what the actual logical record length of this data set is.)

References to Concatenated Data Sets: If you make a **backward reference** to a concatenation (using *.), the system obtains information only from the first data set defined in the sequence of DD statements.

If you make a **forward reference** to a concatenation (using the DDNAME parameter), the system obtains information only from the first data set defined in the sequence of DD statements.

If you issue an assembler **RDJFCB macro instruction** to a data set that is in a concatenation, the system reads the job file control block (JFCB) for only the first data set defined in the sequence of DD statements.

Do Not Concatenate Data Sets to a DUMMY Data Set: If you define a data set using the **DUMMY parameter,** do not concatenate other data sets to it. When the processing program asks to read a dummy data set, the system takes an end-of-data set exit immediately and ignores any data set that might be concatenated to the dummy.

Examples of DD Statements and ddnames

```
//MYDS DD  DSNAME=REPORT
//A    DD  DSNAME=FILE
```

```
//INPUT  DD  DSNAME=FGLIB,DISP=(OLD,PASS)
//       DD  DSNAME=GROUP2,DISP=SHR
```

In this example, because the ddname is missing from the second DD statement, the system concatenates the data sets defined in these statements.

```
//PAYROLL.DAY DD  DSNAME=DESK,DISP=SHR
```

In this example, if procedure step PAYROLL contains a DD statement named DAY, this statement overrides parameters on DD statement DAY. If the step does not contain DD statement DAY, the system adds this statement to procedure step PAYROLL for the duration of the job step.

```
//STEPSIX.DD4 DD  DSNAME=TEXT,DISP=(NEW,PASS)
//            DD  DSNAME=ART,DISP=SHR
```

In this example, the second data set is concatenated to the first, and both are added to procedure step STEPSIX. The ddname is omitted from the second DD statement in order to concatenate data set ART to data set TEXT.

* Parameter

Parameter Type: Positional, optional

Purpose: Use the * parameter to begin an in-stream data set. The data records immediately follow the DD * statement; the records must be in BCD or EBCDIC. The data records end when one of the following is found:

/* in the input stream
// to indicate another JCL statement
The two-character delimiter specified by a DLM parameter on this DD statement
The input stream runs out of card images

Use a DATA parameter instead of the * parameter if any of the data records start with //.

Syntax:

```
//ddname DD *[,parameter]...  [comments]
```

Defaults

When you do not code BLKSIZE and LRECL, JES uses installation defaults specified at initialization.

Relationship to Other Parameters

Restriction When Coding LRECL: If you code LRECL with the * parameter, code a record length of 80 or greater.

Restrictions in a JES2 System: For JES2, the only DD parameters that you can code with the * parameter follow. All other parameters are a JCL error.

DCB	LRECL
DLM	REFDD
DSID	VOLUME
LIKE	

Restrictions in a JES3 System: For JES3, the only DD parameters that you can code with the * parameter follow. All other parameters are a JCL error.

DCB=BLKSIZE	LIKE
DCB=BUFNO	LRECL
DCB=LRECL	REFDD
DCB=MODE=C	VOLUME=SER
DLM	
DSID	

For JES3 SNA RJP Input: The only parameters you can specify for JES3 systems network architecture (SNA) remote job processing (RJP) input devices are BLKSIZE and LRECL.

DD: * Parameter

For 3540 Diskette Input/Output Units: VOLUME=SER, BUFNO, and DSID on a DD * statement are ignored except when they are detected by a diskette reader as a request for an associated data set. See *IBM 3540 Programmer's Reference*. On a DD * or DD DATA statement processed by a diskette reader, you can specify DSID and VOLUME=SER parameters to indicate that a diskette data set is to be merged into the input stream following the DD statement.

Relationship to Other Control Statements

Do not refer to an earlier DD * statement in DCB, DSNAME, or VOLUME parameters on following DD statements.

Location in the JCL

A DD * statement begins an in-stream data set.

In-stream Data for Cataloged or In-stream Procedures: A cataloged or in-stream procedure cannot contain a DD * statement. When you call a procedure, you can add input stream data to a procedure step by placing in the calling step one or more DD * or DD DATA statements, each followed by data.

Multiple In-stream Data Sets for a Step: You can code more than one DD * or DD DATA statement in a job step in order to include several distinct groups of data for the processing program. Precede each group with a DD * or DD DATA statement and follow each group with a delimiter statement. If you omit a DD statement before and a delimiter after input data, the system provides a DD * statement with the ddname of SYSIN and ends the data when it reads a JCL statement or runs out of card images.

Unread Records

If the processing program does not read all the data in an in-stream data set, the system skips the remaining data without abnormally terminating the step.

Examples of the * Parameter

```
//INPUT1   DD  *
           .
           .
           data
           .
//INPUT2   DD  *
           .
           .
           data
           .
/*
```

This example defines two groups of data in the input stream.

```
//STEP2          EXEC PROC=FRESH
//SETUP.WORK     DD   UNIT=3400-6,LABEL=(,NSL)
//SETUP.INPUT1   DD   *
                 .
                 .
                 data
                 .
/*
//PRINT.FRM      DD   UNIT=180
//PRINT.INP      DD   *
                 .
                 .
                 data
                 .
/*
```

This example defines two groups of data in the input stream. The input data defined by DD statement SETUP.INPUT1 is to be used by the cataloged procedure step named SETUP. The input data defined by DD statement PRINT.INP is to be used by the cataloged procedure step named PRINT.

ACCODE Parameter

Parameter Type: Keyword, optional

Purpose: Use the ACCODE parameter to specify or change an accessibility code for an ISO/ANSI/FIPS Version 3 tape output data set. An installation-written file-access exit routine verifies the code after the code is written to tape. If the code is authorized, the job step's program can use the data set; if not, the system issues messages and may abnormally terminate the job step.

A data set protected by an accessibility code should reside only on a volume protected by RACF or a volume accessibility code. The volume should not contain any unprotected data sets.

Note: ACCODE is supported only for ISO/ANSI/FIPS Version 3 tape data sets. ACCODE is ignored for SL (IBM standard) label tapes.

References: For more information on ISO/ANSI/FIPS Version 3 tape data sets, see *Magnetic Tape Labels and File Structure Administration.*

Syntax:

```
ACCODE=access-code
```

Subparameter Definition

access-code
Specifies an accessibility code. The access-code is 1 through 8 characters; the first character is an upper case letter from A through Z.

Note: Only the first character is used as the ISO/ANSI/FIPS Version 3 accessibility code; the other seven characters can be used by the installation. If the first character is other than an upper case letter from A through Z, the installation does not give control to the file-access exit routine.

Defaults

If no accessibility code is specified on a DD statement that defines an ISO/ANSI/FIPS Version 3 tape data set, the system writes a blank character (X'40') in the tape label: a blank authorizes unlimited access to the tape's data sets.

If the installation does not supply a file-access exit routine, the system prevents access to any ISO/ANSI/FIPS Version 3 tape volume.

Overrides

If PASSWORD or NOPWREAD is coded on the DD statement LABEL parameter, password access overrides the ACCODE parameter.

Example of the ACCODE Parameter

```
//TAPE    DD  UNIT=2400,VOLUME=SER=T49850,DSNAME=TAPEDS,
//            LABEL=(,AL),ACCODE=Z
```

In this example, the DD statement ACCODE parameter specifies an accessibility code of Z for tape volume T49850. The volume has ISO/ANSI/FIPS Version 3 labels. The data set TAPEDS is first on the tape.

AMP Parameter

Parameter Type: Keyword, optional

Note: With SMS, you can create new VSAM data sets with JCL DD statements. See the DATACLAS parameter (described on page 10-42) and the RECORG parameter (described on page 10-141).

Purpose: Use the AMP parameter to complete information in an access method control block (ACB) for a VSAM data set. The ACB is a control block for entry-sequenced, key-sequenced, and relative record data sets.

AMP is supported only for VSAM data sets.

References: For more information on VSAM data sets, see *VSAM Administration Guide*, *VSAM Administration: Macro Instruction Reference*, and *JCL User's Guide*.

Syntax:

```
AMP=(subparameter)
AMP=('subparameter[,subparameter]...')

The subparameters are:

      AMORG
      BUFND=number
      BUFNI=number
      BUFSP=number
      CROPS={NCK|NRC|NRE|RCK}
      OPTCD={I|L|IL}
      RECFM={F|FB|V|VB}
      STRNO=number
      SYNAD=module
      TRACE=(subparameter[,subparameter]...)
```

Parentheses: The subparameter or subparameters are always enclosed in one set of parentheses. For example, AMP=(AMORG). Note that AMP='AMORG' is also valid.

Multiple Subparameters: When the parameter contains more than one subparameter, separate the subparameters by commas and enclose the subparameter list in apostrophes inside the parentheses. For example, AMP=('AMORG,STRNO=4').

Special Characters: When the parameter contains only one subparameter and that subparameter contains special characters, enclose the subparameter in apostrophes inside the parentheses. For example, AMP=('STRNO=4').

Note: Do not enclose a subparameter in a subparameter list in apostrophes.

Continuation onto Another Statement: Enclose the subparameter list in only one set of parentheses. Enclose all the subparameters on each statement in apostrophes. End each statement with a comma after a complete subparameter. For example:

```
//DS1   DD  DSNAME=VSAMDATA,AMP=('BUFSP=200,OPTCD=IL,RECFM=FB',
//          'STRNO=6')
```

Subparameter Definition

AMORG

Indicates that the DD statement defines a VSAM data set. Code AMORG for either of the following reasons:

- When data set access is through an ISAM interface program and the DD statement contains VOLUME and UNIT parameters or contains a DUMMY parameter.
- To open an ACB for a VSAM data set, if the data set is not fully defined at the beginning of the job step.

BUFND = number

Specifies the number of I/O buffers that VSAM is to use for data records. The minimum is 1 plus the STRNO subparameter number. This value overrides the BUFND value specified in the ACB or GENCB macro, or provides a value if one is not specified. If you omit STRNO, BUFND must be at least 2.

If you omit BUFND from AMP and from the ACB macro instruction, the system uses the STRNO number plus 1.

BUFNI = number

Specifies the number of I/O buffers that VSAM is to use for index records. This value overrides the BUFNI value specified in the ACB or GENCB macro, or provides a value if one is not specified. If you omit BUFNI from AMP and from the ACB macro instruction, VSAM uses as many index buffers as the STRNO subparameter number; if you omit both BUFNI and STRNO, VSAM uses 1 index buffer.

If data access is through the ISAM interface program, specify for the BUFNI number 1 more than the STRNO number, or specify 2 if you omit STRNO, to simulate having the highest level of an ISAM index resident. Specify a BUFNI number 2 or more greater than the STRNO number to simulate having intermediate levels of the index resident.

BUFSP = number

Specifies the maximum number of bytes for the data and index buffers in the user area. This value overrides the BUFSP value specified in the ACB or GENCB macro, or provides a value if one is not specified.

If BUFSP specifies fewer bytes than the BUFFERSPACE parameter of the access method services DEFINE command, the BUFFERSPACE number overrides the BUFSP number.

CROPS=NCK
CROPS=NRC
CROPS=NRE
CROPS=RCK

Requests a checkpoint/restart option. For more information, see *Checkpoint/Restart User's Guide*.

NCK
Requests no data set post-checkpoint modification tests.

NRC
Requests neither a data-erase test nor data set post-checkpoint modification tests.

NRE
Requests no data-erase test.

RCK
Requests a data-erase test and data set post-checkpoint modification tests. If the CROPS subparameter is omitted, RCK is the default.

If you request an inappropriate option, such as the data-erase test for an input data set, the system ignores the option.

OPTCD=I
OPTCD=L
OPTCD=IL

Indicates how the ISAM interface program is to process records that the step's processing program flags for deletion.

I
Requests, when the data control block (DCB) contains OPTCD=L, that the ISAM interface program is not to write into the data set records marked for deletion by the processing program.

If AMP=('OPTCD=I') is specified without OPTCD=L in the DCB, the system ignores deletion flags on records.

L
Requests that the ISAM interface program is to keep in the data set records marked for deletion by the processing program.

If records marked for deletion are to be kept but OPTCD=L is not in the DCB, AMP=('OPTCD=L') is required.

Note: This parameter has the same meaning and restrictions for the ISAM interface as it has for ISAM. While it was not required in the ISAM job control language, you should code it in the AMP parameter.

IL
Requests that the ISAM interface program is not to write into the data set records marked for deletion by the processing program. If the processing program had read the record for update, the ISAM interface program deletes the record from the data set.

AMP=('OPTCD=IL') has the same effect as AMP=('OPTCD=I') coded with OPTCD=L in the DCB.

RECFM=F
RECFM=FB
RECFM=V
RECFM=VB

(For data sets with SMS, see the DD RECFM parameter described on page 10-136.)

Identifies the ISAM record format used by the processing program. You must code this RECFM subparameter when the record format is not specified in the DCB.

Note: This parameter has the same meaning and restrictions for the ISAM interface as it has for ISAM. While it was not required in the ISAM job control language, you should code it in the AMP parameter.

All VSAM requests are for unblocked records. If the processing program requests blocked records, the ISAM interface program sets the overflow-record indicator for each record to indicate that each is being passed to the program unblocked.

F
Indicates fixed-length records.

FB
Indicates blocked fixed-length records.

V
Indicates variable-length records. If no RECFM is specified in the AMP parameter or in the DCB, V is the default.

VB
Indicates blocked variable-length records.

STRNO=number
Indicates the number of request parameter lists the processing program uses concurrently. The number must at least equal the number of BISAM and QISAM requests that the program can issue concurrently. If the program creates subtasks, add together the number of requests for each subtask plus 1 for each subtask that sequentially processes the data set. This value overrides the STRNO value specified in the ACB or GENCB macro, or provides a value if one is not specified.

SYNAD=module
Names a SYNAD exit routine. The ISAM interface program is to load and exit to this routine if a physical or logical error occurs when the processing program is gaining access to the data set.

The SYNAD parameter overrides a SYNAD exit routine specified in the EXLST or GENCB macro instruction that generates the exit list. The address of the intended exit list is specified in the access method control block that links this DD statement to the processing program. If no SYNAD exit is specified, the system ignores the AMP SYNAD parameter.

TRACE = (subparameter[,subparameter]...)

Indicates that the generalized trace facility (GTF) executes with your job to gather information about the opening, closing, and end-of-volume processing for the data set defined on this DD statement. You can use the AMDPRDMP program to print the trace output; see *SPL: Service Aids.*

The TRACE subparameters are: HOOK, ECODE, KEY, PARM1, and PARM2. See *VSAM Administration Guide* for full information on the TRACE subparameter and the VSAM trace facility, which you use to obtain diagnostic information during VSAM processing.

Relationship to Other Parameters

Do not code the following parameters with the AMP parameter.

*	DDNAME	RECFM
BURST	DYNAM	SUBSYS
CHARS	FCB	SYSOUT
COPIES	FLASH	TERM
DATA	MODIFY	UCS
DCB	QNAME	

Invalid ddnames: The following ddnames are invalid for VSAM data sets:

JOBLIB
STEPLIB
SYSABEND
SYSCHK
SYSCKEOV
SYSMDUMP
SYSUDUMP

Invalid DSNAMEs: When you code the AMP parameter, the DSNAME must not contain parentheses, a minus (hyphen), or a plus (+) sign. The forms of DSNAME valid for ISAM, partitioned access method (PAM), and generation data groups (GDG) are invalid with VSAM data sets.

Buffer Requirements

For a key-sequenced data set, the total minimum buffer requirement is three: two data buffers and one index buffer. For an entry-sequenced data set, two data buffers are required.

If the number of buffers specified in the BUFND and BUFNI subparameters causes the virtual storage requirements to exceed the BUFSP space, the number of buffers is reduced to fit in the BUFSP space.

If BUFSP specifies more space than required by BUFND and BUFNI, the number of buffers is increased to fill the BUFSP space.

Examples of the AMP Parameter

```
//VSAMDS1  DD  DSNAME=DSM.CLASS,DISP=SHR,AMP=('BUFSP=200,BUFND=2',
//             'BUFNI=3,STRNO=4,SYNAD=ERROR')
```

In this example, the DD statement defines the size of the user area for data and index buffers, specifies the number of data and index buffers, specifies the number of requests that require concurrent data set positioning, and specifies an error exit routine named ERROR.

```
//VSAMDS2  DD  DSNAME=DSM.CLASS,DISP=SHR,AMP=('BUFSP=23456,BUFND=5',
//             'BUFNI=10,STRNO=6,SYNAD=ERROR2,CROPS=NCK',
//             'TRACE=(PARM1=F00203000010,KEY=ABCDEF)')
```

In this example, the DD statement defines the values for BUFSP, BUFNI, STRNO, and SYNAD, as in the previous example. It also specifies that a data set post-checkpoint modification test is not to be performed when restarting at a checkpoint and that GTF is to provide a trace of specified data areas.

AVGREC Parameter

Parameter Type: Keyword, optional — use this parameter only with SMS

Purpose: Use the AVGREC parameter when you define a new data set to specify that:

- The units of allocation requested for storage space are records.
- The primary and secondary space quantity specified on the SPACE parameter represents units, thousands, or millions of records.

When you use AVGREC with the SPACE parameter, the first subparameter (reclgth) on the SPACE parameter must specify the average record length of the records.

Code the AVGREC parameter when you want to (1) specify records as the units of allocation or (2) override the space allocation defined in the data class for the data set.

If SMS is not installed or is not active, the system syntax checks and then ignores the AVGREC parameter.

Syntax:

```
AVGREC={U}
       {K}
       {M}
```

Subparameter Definition

U

Specifies a record request and that the primary and secondary space quantity specified on the SPACE parameter represents the number of records in units (multiplier of 1).

K

Specifies a record request and that the primary and secondary space quantity specified on the SPACE parameter represents the number of records in thousands (multiplier of 1024).

M

Specifies a record request and that the primary and secondary space quantity specified on the SPACE parameter represents the number of records in millions (multiplier of 1048576).

Overrides

AVGREC overrides the space allocation defined in the data class for the data set.

Relationship to Other Parameters

Do not code AVGREC with the TRK, CYL, or ABSTR subparameters of the SPACE parameter.

Do not code the following DD parameters with the AVGREC parameter.

*	DYNAM
DATA	QNAME
DDNAME	

Examples of the AVGREC Parameter

```
//SMSDS3   DD  DSNAME=MYDS3.PGM,DATACLAS=DCLAS03,DISP=(NEW,KEEP),
//             SPACE=(128,(5,2)),AVGREC=K
```

In the example, the space allocation defined in the DCLAS03 data class is overridden by the SPACE and AVGREC parameters, which indicate an average record length of 128 bytes, a primary quantity of 5K (5,120) records, and a secondary quantity of 2K (2,048) records.

```
//SMSDS3A DD  DSNAME=MYDS3.PGM,DATACLAS=DCLAS03A,DISP=(NEW,KEEP),
//            AVGREC=K
```

In the example, the space allocation defined in the DCLAS03A data class is overridden by the AVGREC parameter, which indicates that the primary and secondary quantity represents thousands of records.

BURST Parameter

Parameter Type: Keyword, optional

Purpose: Use the BURST parameter to specify that the output for this sysout data set printed on a 3800 Printing Subsystem is to go to:

- The burster-trimmer-stacker, to be burst into separate sheets.
- The continuous forms stacker, to be left in continuous fanfold.

If the specified stacker is different from the last stacker used, or if a stacker was not previously requested, JES issues a message to the operator to thread the paper into the required stacker.

Note: BURST applies only for an output data set printed on a 3800 equipped with a burster-trimmer-stacker.

Syntax:

```
BURST=(YES )
      (Y   )
      (NO  )
      (N   )
```

Subparameter Definition

YES
Requests that the printed output is to be burst into separate sheets. This subparameter can also be coded as Y.

NO
Requests that the printed output is to be in a continuous fanfold. This subparameter can also be coded as N.

Defaults

If you do not code a BURST parameter, but you code a DD SYSOUT parameter and the sysout data set is printed on a 3800 that has a burster-trimmer-stacker, JES uses an installation default specified at initialization.

If you do not code a BURST parameter or a DD SYSOUT parameter, the default is NO.

Overrides

A BURST parameter on a sysout DD statement overrides an OUTPUT JCL BURST parameter.

Relationship to Other Parameters

Do not code the following parameters with the BURST parameter.

*	DSID	PROTECT
AMP	DSNAME	QNAME
DATA	DYNAM	SUBSYS
DDNAME	LABEL	VOLUME
DISP	MSVGP	

Relationship to Other Control Statements

The burster-trimmer-stacker can also be requested using the following:

- The BURST parameter on the OUTPUT JCL statement.
- The STACKER parameter on the JES3 //*FORMAT PR statement.
- The BURST parameter on the JES2 /*OUTPUT statement.

Example of the BURST Parameter

```
//RECORD DD SYSOUT=A,BURST=Y
```

In this example, the DD statement requests that JES send the output to the burster-trimmer-stacker of the 3800. The stacker separates the printed output into separate sheets instead of stacking it in a continuous fanfold.

CHARS Parameter

Parameter Type: Keyword, optional

Purpose: Use the CHARS parameter to specify the name of one or more character-arrangement tables for printing this sysout data set on a 3800 Printing Subsystem.

Note: CHARS applies only for an output data set that is printed on a 3800.

References: For more information on character-arrangement tables, see the *3800 Printing Subsystem Programmer's Guide*. Refer to *Installation: System Generation* for information on how to choose during system generation particular groups, other than the Basic group, which is always available.

Syntax:

```
      ┌table-name                          ┐
CHARS=│(table-name[,table-name]...)        │
      │DUMP                                │
      └(DUMP[,table-name]...)              ┘
```

- You can omit the parentheses if you code only one table-name or only DUMP.
- Null positions in the CHARS parameter are invalid. For example, you **cannot** code CHARS=(,table-name) or CHARS=(table-name,,table-name).

Subparameter Definition

table-name
Names a character-arrangement table. Each table-name is 1 through 4 alphanumeric or national ($, #, @) characters. Code from one to four names.

DUMP
Requests a high-density dump of 204-character print lines from a 3800. If more than one table-name is coded, DUMP must be first.

Note: Use DUMP on a SYSABEND or SYSUDUMP DD statement.

Defaults

If you do not code the DD CHARS parameter, JES uses the following, in order:

1. The CHARS parameter on an OUTPUT JCL statement, if referenced by the DD statement.
2. The DD UCS parameter value, if coded.
3. The UCS parameter on an OUTPUT JCL statement, if referenced.

If no character-arrangement table is specified on the DD or OUTPUT JCL statements, JES uses an installation default specified at initialization.

Overrides

A CHARS parameters on a sysout DD statement overrides the OUTPUT JCL CHARS parameter.

For a data set scheduled to the Print Services Facility (PSF), the PSF uses the following parameters, in override order, to select the font list:

1. Font list in the library member specified by an OUTPUT JCL PAGEDEF parameter.
2. DD CHARS parameter.
3. OUTPUT JCL CHARS parameter.
4. DD UCS parameter.
5. OUTPUT JCL UCS parameter.
6. JES installation default for the device.
7. Font list on the PAGEDEF parameter in the PSF cataloged procedure.

See "PAGEDEF Parameter" on page 17-51 for more information.

Relationship to Other Parameters

Do not code the following parameters with the CHARS parameter.

*	DSID	PROTECT
AMP	DSNAME	QNAME
DATA	DYNAM	SUBSYS
DDNAME	LABEL	VOLUME
DISP	MSVGP	

DD: CHARS

Relationship to Other Control Statements

CHARS can also be coded on the following:

- The OUTPUT JCL statement.
- The JES3 //*FORMAT PR statement.
- The JES2 /*OUTPUT statement.

Printing Device Reassignment

The output device might not be a 3800, for example, if printing were reassigned to a 3211. See the *3800 Printing Subsystem Programmer's Guide* for restrictions that apply.

Requesting a High-Density Dump

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 of the CHARS Parameter

```
//DD1  DD  SYSOUT=A,CHARS=(GS10,GU12)
```

In this example, the CHARS parameter specifies two character-arrangement tables to be used when printing the data set: GS10 and GU12.

```
//SYSABEND DD  UNIT=3800,CHARS=DUMP,FCB=STD3
```

The CHARS parameter on this SYSABEND DD statement specifies a high-density dump with 204 characters per line. The FCB parameter requests the dump output at 8 lines per inch.

CHKPT Parameter

Parameter Type: Keyword, optional

Purpose: Use the CHKPT parameter to request that a checkpoint be written when each end-of-volume is reached on the multivolume data set defined by this DD statement. Checkpoints are written for all volumes except the last. Checkpoints can be requested for input or output data sets.

Note: CHKPT is supported only for multivolume QSAM or BSAM data sets. CHKPT is ignored for single-volume QSAM or BSAM data sets or for ISAM, BDAM, BPAM, or VSAM data sets.

References For more information, see *Checkpoint/Restart User's Guide.*

Syntax:

```
CHKPT=EOV
```

Subparameter Definition

EOV
Requests a checkpoint at each end-of-volume.

Overrides

- The RD parameter values of NC and RNC on the JOB or EXEC statements override the CHKPT parameter.
- The CHKPT parameter overrides cataloged procedure values or START command values for checkpoints at end-of-volume.

Relationship to Other Parameters

Do not code the following parameters with the CHKPT parameter.

*	DYNAM
DATA	QNAME
DDNAME	SYSOUT

Relationship to the SYSCKEOV DD Statement

If you specify CHKPT, you must also provide a SYSCKEOV DD statement in the job or step.

DD: CHKPT

Checkpointing Concatenated Data Sets

For concatenated BSAM or QSAM data sets, CHKPT must be coded on each DD statement in the concatenation, if checkpointing is desired for each data set in the concatenation.

Examples of the CHKPT Parameter

```
//DS1 DD  DSNAME=INDS,DISP=OLD,CHKPT=EOV,
//        UNIT=SYSSQ,VOLUME=SER=(TAPE01,TAPE02,TAPE03)
```

In this example, the DD statement defines data set INDS, a multivolume QSAM or BSAM data set for which a checkpoint is to be written twice: once when end-of-volume is reached on TAPE01 and once when end-of-volume is reached on TAPE02.

```
//DS2 DD  DSNAME=OUTDS,DISP=(NEW,KEEP),
//        CHKPT=EOV,UNIT=SYSDA,VOLUME=(,,,8)
```

In this example, OUTDS is a multivolume data set that is being created. The data set requires eight volumes. Seven checkpoints will be written: when the end-of-volume is reached on volumes one through seven.

CNTL Parameter

Parameter Type: Keyword, optional

Purpose: Use the CNTL parameter to reference a CNTL statement that appears earlier in the job. The reference causes the subsystem to execute the program control statements within the referenced CNTL/ENDCNTL group.

The system searches for an earlier CNTL statement with a label that matches the **label** in the CNTL parameter. If the system finds no match, the system issues an error message.

Syntax:

```
CNTL={*.label                          }
     {*.stepname.label                 }
     {*.stepname.procstepname.label    }
```

Subparameter Definition

***.label**
Identifies an earlier CNTL statement, named label. The system searches for the CNTL statement first earlier in this step, then before the first EXEC statement of the job.

***.stepname.label**
Identifies an earlier CNTL statement, named label, that appears in an earlier step, stepname, in the same job.

***.stepname.procstepname.label**
Identifies a CNTL statement, named label, in a cataloged or in-stream procedure. Stepname is the name of the job step that calls the procedure; procstepname is the name of the procedure step that contains the CNTL statement named label.

Examples of the CNTL Parameter

```
//MONDAY    DD   CNTL=*.WKLYPGM
```

In this example, the DD statement requests that the system use the program control statements following the CNTL statement named WKLYPGM and located earlier in this step or preceding the first step.

```
//TUESDAY   DD   CNTL=*.SECOND.BLOCKS
```

In this example, the DD statement requests that the system use the program control statements following the CNTL statement named BLOCKS and located in a preceding step named SECOND.

```
//WEDNES    DD   CNTL=*.THIRD.PROCTWO.CANETTI
```

In this example, the DD statement requests that the system use the program control statements following the CNTL statement named CANETTI and located in the procedure step PROCTWO of the procedure called in the preceding job step THIRD.

COPIES Parameter

Parameter Type: Keyword, optional

Purpose: Use the COPIES parameter to specify how many copies of this sysout data set are to be printed. The printed output is in page sequence for each copy.

For printing on a 3800 Printing Subsystem, this parameter can instead specify how many copies of each page are to be printed before the next page is printed.

Syntax:

```
COPIES={nnn                                          }
       {(nnn,(group-value[,group-value]...))         }
       {(,(group-value[,group-value]...))            }
```

- You can omit the parentheses if you code only COPIES=nnn.
- The following are **not** valid:
  - A null group-value, for example, COPIES=(5,(,)) or COPIES=(5,)
  - A zero group-value, for example, COPIES=(5,(1,0,4))
  - A null within a list of group-values, for example, COPIES=(5,(1,,4))

Subparameter Definition

nnn

Specifies how many copies of the data set are to be printed; each copy will be in page sequence order. nnn is a number from 1 through 255 in a JES2 system and from 1 through 254 in a JES3 system

For a data set printed on a 3800, JES ignores nnn if any group-values are specified.

group-value

Specifies how many copies of each page are to be printed before the next page is printed. Each group-value is a number from 1 through 255 in a JES2 system and from 1 through 254 in a JES3 system. You can code a maximum of eight group-values. Their sum must not exceed 255 or 254. The total copies of each page equals the sum of the group-values.

Note:

- This subparameter is valid only for 3800 output.
- For 3800 output, this subparameter overrides an nnn subparameter, if coded.

DD: COPIES

Defaults

If you do not code a COPIES parameter on any of the following, code it incorrectly, or code COPIES=0, the system uses a default of 1, which is the default for the DD COPIES parameter.

DD statement
OUTPUT JCL statement
For JES2, the /*OUTPUT statement

Overrides

A COPIES parameter on a sysout DD statement overrides an OUTPUT JCL COPIES parameter.

If this DD statement references an OUTPUT JCL statement and that OUTPUT JCL statement contains a FORMDEF parameter, which specifies a library member, the COPYGROUP parameter on a FORMDEF statement in that member overrides any group-value subparameters on the OUTPUT JCL COPIES parameter or the sysout DD COPIES parameter. For more information, see "FORMDEF Parameter" on page 17-38.

Relationship to Other Parameters

Do not code the following parameters with the COPIES parameter.

*	DSNAME	QNAME
AMP	DYNAM	SUBSYS
DATA	LABEL	VOLUME
DDNAME	MSVGP	
DISP		

Relationship to FLASH Parameter: If this DD statement or a referenced OUTPUT JCL statement also contains a FLASH parameter, JES prints with the forms overlay the number of copies specified in one of the following:

- COPIES=nnn, if the FLASH count is larger than nnn. For example, if COPIES=10 and FLASH=(LTHD,12) JES prints 10 copies, all with the forms overlay.
- The sum of the group-values specified in the COPIES parameter, if the FLASH count is larger than the sum. For example, if COPIES=(,(2,3,4)) and FLASH=(LTHD,12) JES prints nine copies in groups, all with the forms overlay.
- The count subparameter in the FLASH parameter, if the FLASH count is smaller than nnn or the sum from the COPIES parameter. For example, if COPIES=10 and FLASH=(LTHD,7) JES prints seven copies with the forms overlay and three copies without.

Restriction When Coding UNIT Parameter: The COPIES parameter is normally coded with the SYSOUT parameter. If, however, both COPIES and UNIT appear on a DD statement, JES handles the COPIES parameter as follows:

- nnn defaults to 1.
- Only the first group-value is used, if group-values are specified and printing is on a 3800.

Relationship to Other Control Statements

The number of copies can also be specified on the COPIES parameter of the following:

- The OUTPUT JCL statement.
- The JES2 /*OUTPUT statement.
- The JES3 //*FORMAT PR statement.
- The JES3 //*FORMAT PU statement.

For JES2, if you request copies of the entire job on the JES2 /*JOBPARM COPIES parameter and also copies of the data set on the DD COPIES or OUTPUT JCL COPIES parameter, and if this is a sysout data set, JES2 prints the number of copies equal to the **product** of the two requests.

Using OUTPUT JCL COPIES by Nullifying DD Copies

If both a DD statement and a referenced OUTPUT JCL statement contain COPIES parameters, the DD COPIES parameter normally overrides the OUTPUT JCL COPIES parameter. For example, four copies are printed of sysout data set DDA:

```
//OTA  OUTPUT  COPIES=3
//DDA  DD      SYSOUT=A,OUTPUT=*.OTA,COPIES=4
```

However, if the DD COPIES is a null parameter, the OUTPUT JCL COPIES parameter is used. For example, three copies are printed of sysout data set DDB:

```
//OTB  OUTPUT  COPIES=3
//DDB  DD      SYSOUT=A,OUTPUT=*.OTB,COPIES=
```

The following example shows a null COPIES parameter on an in-stream DD statement that overrides a procedure DD statement. The null COPIES parameter on DD statement PS.DDA nullifies the COPIES parameter on the procedure DD statement DDA, thereby allowing the COPIES parameter on OUTPUT JCL statement OT to be used. The system prints three copies of the DDA sysout data set.

```
//JEX      JOB     ACCT34,'PAUL BENNETT'
//INSTR    PROC
//PS       EXEC    PGM=ABC
//OT       OUTPUT  COPIES=3
//DDA      DD      SYSOUT=A,OUTPUT=*.OT,COPIES=2
//         PEND
//STEP1    EXEC    PROC=INSTR
//PS.DDA   DD      COPIES=
/*
```

Note: If a null COPIES parameter appears on a DD statement that either does not reference an OUTPUT JCL statement or references an OUTPUT JCL statement that does not contain a COPIES parameter, the system uses a default of 1.

Examples of the COPIES Parameter

```
//RECORD1 DD  SYSOUT=A,COPIES=32
```

This example requests 32 copies of the data set defined by DD statement RECORD1 when printing on an impact printer or a 3800.

```
//RECORD2 DD  SYSOUT=A,COPIES=(0,(1,2))
```

In this example, when printing on a 3800, three copies of the data set are printed in two groups. The first group contains one copy of each page. The second group contains two copies of each page. When printing on an impact printer, one copy (the default for nnn) is printed.

```
//RECORD3 DD  SYSOUT=A,COPIES=(8,(1,3,2))
```

In this example, when printing on a 3800, six copies of the data set are printed in three groups. The first group contains one copy of each page, the second group contains three copies of each page, and the last group contains two copies of each page. When the output device is not a 3800, the system prints eight collated copies.

```
//RECORD4 DD  UNIT=3800,COPIES=(1,(2,3))
```

Because the UNIT parameter is coded and the device is a 3800, the system prints only the first group-value: two copies of each page.

DATA Parameter

Parameter Type: Positional, optional

Purpose: Use the DATA parameter to begin an in-stream data set that contains statements with // in columns 1 and 2. The data records immediately follow the DD DATA statement; the records must be in BCD or EBCDIC. The data records end when one of the following is found:

- /* in the input stream
- The two-character delimiter specified by a DLM parameter on this DD statement
- The input stream runs out of card images

Note that, unlike a DD * statement, the data is not ended by the // that indicates another JCL statement.

Syntax:

```
//ddname DD DATA[,parameter]...  [comments]
```

Defaults

When you do not code BLKSIZE and LRECL, JES uses installation defaults specified at initialization.

Relationship to Other Parameters

Restrictions in a JES2 System: For JES2, the only DD parameters that you can code with the DATA parameter follow. All other parameters are a JCL error.

DCB	LRECL
DLM	REFDD
DSID	VOLUME
LIKE	

Restrictions in a JES3 System: For JES3, the only DD parameters that you can code with the DATA parameter follow. All other parameters are a JCL error.

DCB=BLKSIZE	LIKE
DCB=BUFNO	LRECL
DCB=LRECL	REFDD
DCB=MODE=C	VOLUME=SER
DLM	
DSID	

For JES3, when using the DCB=MODE=C subparameter with the DATA parameter, DCB=MODE=C must be the only parameter specified with the DATA parameter.

For 3540 Diskette Input/Output Units: VOLUME=SER, BUFNO, and DSID on a DD DATA statement are ignored except when they are detected by a diskette reader as a request for an associated data set. See *IBM 3540 Programmer's Reference*. On a DD * or DD DATA statement processed by a diskette reader, you can specify DSID and VOLUME=SER parameters to indicate that a diskette data set is to be merged into the input stream following the DD statement.

Relationship to Other Control Statements

Do not refer to an earlier DD DATA statement in DCB, DSNAME, or VOLUME parameters on following DD statements.

Location in the JCL

A DD DATA statement begins an in-stream data set.

In-stream Data for Cataloged or In-stream Procedures: A cataloged or in-stream procedure cannot contain a DD DATA statement. When you call a procedure, you can add input stream data to a procedure step by placing in the calling step one or more DD * or DD DATA statements, each followed by data.

Multiple In-stream Data Sets for a Step: You can code more than one DD * or DD DATA statement in a job step in order to include several distinct groups of data for the processing program. Precede each group with a DD * or DD DATA statement and follow each group with a delimiter statement. If you omit a DD statement before and a delimiter after input data, the system provides a DD * statement with the ddname of SYSIN and ends the data when it reads a JCL statement or runs out of card images.

Unread Records

If the processing program does not read all the data in an in-stream data set, the system skips the remaining data without abnormally terminating the step.

Examples of the DATA Parameter

```
//GROUP1   DD  DATA
           .
           .
           data
           .
/*
//GROUP2   DD  DATA
           .
           .
           data
           .
/*
```

This example defines two groups of data in the input stream.

```
//STEP2       EXEC PROC=UPDATE
//PREP.DD4    DD   DSNAME=A.B.C,UNIT=3350,VOLUME=SER=D88230,
//               SPACE=(TRK,(10,5)),DISP=(,CATLG,DELETE)
//PREP.IN1    DD   DATA
                .
                .
                data
                .
/*
//ADD.IN2     DD   *
                .
                .
                data
                .
/*
```

This example defines two groups of data in the input stream. The input defined by DD statement PREP.IN1 is to be used by the cataloged procedure step named PREP. This data contains job control statements. The input data defined by DD statement ADD.IN2 is to be used by the cataloged procedure step named ADD. Because this data is defined by a DD * statement, it must not contain job control statements.

DATACLAS Parameter

Parameter Type: Keyword, optional — use this parameter only with SMS

Without SMS, use the DCB parameter (described on page 10-44) or the AMP parameter (described on page 10-18).

Purpose: Use the DATACLAS parameter to specify a data class for a new data set. The storage administrator at your installation defines the names of the data classes you can code on the DATACLAS parameter.

If SMS is not installed or is not active, the system syntax checks and then ignores the DATACLAS parameter.

SMS ignores the DATACLAS parameter if you specify it for (1) an existing data set or (2) a data set that SMS does not support (such as a tape data set).

You can use the DATACLAS parameter for both VSAM data sets and physical sequential (PS) or partitioned (PO) data sets.

A data class defines the following data set allocation attributes:

- Data set organization
  - Record organization (RECORG) or
  - Record format (RECFM)
- Record length (LRECL)
- Key length (KEYLEN)
- Key offset (KEYOFF)
- Space allocation (AVGREC and SPACE)
- Retention period (RETPD) or expiration date (EXPDT)
- Volume-count (VOLUME)
- For VSAM data sets (IMBED or REPLICATE, CISIZE, FREESPACE, SHAREOPTIONS)

References: See *MVS/XA Interactive Storage Management Facility User's Guide* for information on how to use ISMF to view your installation-defined data classes.

Syntax:

```
DATACLAS=data-class-name
```

Subparameter Definition

data-class-name
Specifies the name of a data class to be used for allocating the data set.

The name, one to eight characters, is defined by the storage administrator at your installation.

Defaults

If you do not specify DATACLAS for a new data set and the storage administrator has provided an installation-written automatic class selection (ACS) routine, the ACS routine may select a data class for the data set. Check with your storage administrator to determine if an ACS routine will select a data class for the new data set, in which case you do not need to specify DATACLAS.

Overrides

Any attributes you specify on the same DD statement using the following parameters, override the corresponding attributes in the named data class for the data set:

RECORG (record organization) or RECFM (record format)
LRECL (record length)
KEYLEN (key length)
KEYOFF (key offset)
AVGREC (record request and space quantity)
SPACE (average record length, primary, secondary, and directory quantity)
RETPD (retention period) or EXPDT (expiration date)
VOLUME (volume-count)

An ACS routine can override the data class that you specify on the DATACLAS parameter.

Relationship to Other Parameters

Do not code the following DD parameters with the DATACLAS parameter.

*	DYNAM
DATA	QNAME
DDNAME	

Examples of the DATACLAS Parameter

```
//SMSDS1  DD  DSNAME=MYDS1.PGM,DATACLAS=DCLAS01,DISP=(NEW,KEEP)
```

In the example, the attributes in the data class named DCLAS01 are used by SMS to handle the data set. Note that installation-written ACS routines may select a management class and storage class and can override the specified data class.

```
//SMSDS2  DD  DSNAME=MYDS2.PGM,DATACLAS=DCLAS02,DISP=(NEW,KEEP),
//              LRECL=256,EXPDT=1996/033
```

In the example, the logical record length of 256 and the expiration date of February 2, 1996, override the corresponding attributes defined in the data class for the data set. Note that installation-written ACS routines may select a management class and storage class and can override the specified data class.

DCB Parameter

Parameter Type: Keyword, optional

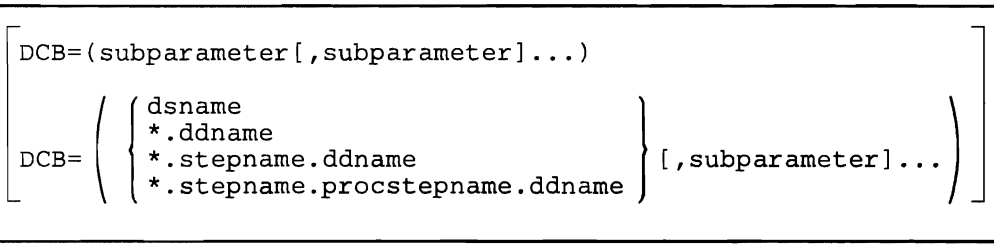
Note: With SMS, you do not need to use the DCB parameter to specify data set attributes. See the DATACLAS parameter (described on page 10-42), the LIKE parameter (described on page 10-115), and the REFDD parameter (described on page 10-143).

Purpose: Use the DCB parameter to complete during execution the information in the data control block (DCB) for a data set.

The data control block is constructed by the DCB macro instruction in assembler language programs or by file definition statements or language-defined defaults in programs in other languages.

References: For more information on constructing the data control block, see *Data Administration Guide.*

Syntax:



Parentheses: You can omit the parentheses if you code:

- Only one keyword subparameter.
- Only a data set name, dsname, without any subparameters.
- Only a backward reference without any subparameters. A backward reference is a reference to an earlier DD statement in the job or in a cataloged or in-stream procedure called by a job step. A backward reference is in the form *.ddname or *.stepname.ddname or *.stepname.procstepname.ddname.

For example, DCB=RECFM=FB or DCB=WKDATA or DCB=*.STEP3.DD2

Multiple Subparameters: When the parameter contains more than one subparameter, separate the subparameters by commas and enclose the subparameter list in parentheses. For example, DCB=(RECFM=FB,LRECL=133,BLKSIZE=399) or DCB=(*.DD1,BUFNO=4)

Continuation onto Another Statement: Enclose the subparameter list in only one set of parentheses. End each statement with a comma after a complete subparameter. For example:

```
//INPUT  DD  DSNAME=WKDATA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,
//            BUFL=800,BUFNO=4)
```

Alternate Syntax for DCB Keyword Subparameters:

All of the DCB keyword subparameters can be specified without the need to code DCB=. For example, the following DD statement:

```
//DDEX   DD   DSNAME=WKDATA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),DISP=MOD
```

can also be specified as:

```
//DDEX   DD   DSNAME=WKDATA,RECFM=FB,LRECL=80,BLKSIZE=800,DISP=MOD
```

Note that KEYLEN, LRECL, and RECFM are described as DD parameters.

Subparameter Definition

subparameter
(With SMS, see the DD DATACLAS parameter.)

Specifies a DCB keyword subparameter needed to complete the data control block.

An alphabetic summary of the DCB keyword subparameters follows this parameter description.

You must supply DCB information through the DCB subparameters if your processing program, the data set label, or your language's defined values do not complete the data control block.

dsname
(With SMS, see the DD LIKE parameter.)

Names a cataloged data set. The system is to copy DCB information from the data set's label. The data set must reside on a direct access volume, and the volume must be mounted before the job step is executed.

A hyphen is a valid character in a catalogued data set name. A data set name that contains a hyphen must be enclosed in apostrophes if it is used as a DCB subparameter.

The dsname cannot contain any other special characters, except for periods used in qualifying the name. Do not specify a generation data group (GDG) name.

The system copies the following DCB information from the data set label:

DSORG (used in a backward reference)
RECFM
OPTCD
BLKSIZE
LRECL
KEYLEN
RKP

If you do not specify the system code and expiration date of the cataloged data set, the system copies them from the data set label.

If you code any DCB subparameters after the dsname, these subparameters override the corresponding subparameters in the data set label. The system copies from the referenced label only those subparameters not specified on the referencing DD statement.

***.ddname**
***.stepname.ddname**
***.stepname.procstepname.ddname**
(With SMS, see the DD REFDD parameter.)

Specify a backward reference to an earlier DD statement. The system is to copy DCB information from the DCB parameter specified on that DD statement. The DCB parameter of the referenced DD statement must contain subparameters, and it cannot name a cataloged data set or refer to another DD statement.

*.ddname specifies the ddname of an earlier DD statement in the same step. *.stepname.ddname specifies the ddname of a DD statement in an earlier step, stepname, in the same job. *.stepname.procstepname.ddname specifies the ddname of a DD statement in a cataloged or in-stream procedure called by an earlier job step. Stepname is the name of the job step that calls the procedure and procstepname is the name of the procedure step that contains the DD statement.

If you code any DCB subparameters after the reference, these subparameters override the corresponding subparameters on the referenced DD statement. The system copies from the referenced DD statement only those subparameters not specified on the referencing DD statement.

Do not reference a DD * or a DD DATA statement.

Note: The system also copies the UCS and FCB parameters from the referenced DD statement, unless you override them in the referencing DD statement.

Completing the Data Control Block

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

- The processing program, that is, the DCB macro instruction in assembler language programs or file definition statements or language-defined defaults in programs in other languages.
- The DCB subparameter of the DD statement.
- The data set label.

Therefore, if you supply information for the same DCB field in your processing program and on a DD statement, the system ignores the DD DCB subparameter. If a DD statement and the data set label supply information for the same DCB field, the system ignores the data set label information.

Note: When concatenated data sets are involved, the DCB is completed based on the type of data set and how the processing program uses the data set. See *MVS/XA Data Administration Guide* for more information.

Relationship to Other Parameters

See the descriptions of the individual DCB subparameters for the DD parameters and DCB subparameters that should not be coded with a specific DCB subparameter.

Do not code the following parameters with the DCB parameter.

AMP
DYNAM

With the DDNAME parameter, code **only** the BLKSIZE, BUFNO, and DIAGNS DCB subparameters.

With the QNAME parameter, code **only** the BLKSIZE, LRECL, OPTCD, and RECFM DCB subparameters.

The DD parameter KEYLEN and DCB subparameters KEYLEN, MODE, PRTSP, STACK, and TRTCH apply to specific device types. If you specify one of these subparameters on a DD statement for a device different from the type to which it applies, the system interprets the value incorrectly.

For 3540 Diskette Input/Output Units: The VOLUME=SER, DCB=BUFNO, and DSID parameters on a DD * or DD DATA statement are ignored except when they are detected by a diskette reader as a request for an associated data set. See *IBM 3540 Programmer's Reference*.

Examples of the DCB Parameter

```
//DD1   DD    DSNAME=ALP,DISP=(,KEEP),VOLUME=SER=44321,
//            UNIT=3400-6,DCB=(RECFM=FB,LRECL=240,BLKSIZE=960,
//            DEN=1,TRTCH=C)
```

DD statement DD1 defines a new data set named ALP. The DCB parameter contains the information necessary to complete the data control block.

```
//DD1A  DD    DSNAME=EVER,DISP=(NEW,KEEP),UNIT=3380,
//            DCB=(RECFM=FB,LRECL=326,BLKSIZE=23472),
//            SPACE=(23472,(200,40))
```

DD statement DD1A defines a new data set named EVER on a 3380. The DCB parameter contains the information necessary to complete the data control block.

```
//DD1B  DD    DSNAME=EVER,DISP=(NEW,KEEP),UNIT=3380,
//            RECFM=FB,LRECL=326,
//            SPACE=(23472,(200,40))
```

DD statement DD1B is the same as the DD1A statement except that it shows the alternate syntax for the DCB keyword subparameters. Also, because BLKSIZE is omitted, the system will select an optimum block size for the data.

```
//DD2    DD     DSNAME=BAL,DISP=OLD,DCB=(RECFM=F,LRECL=80,
//             BLKSIZE=80)
//DD3    DD     DSNAME=CNANN,DISP=(,CATLG,DELETE),UNIT=3400-6,
//             LABEL=(,NL),VOLUME=SER=663488,DCB=*.DD2
```

DD statement DD3 defines a new data set named CNANN and requests that the system copy the DCB subparameters from DD statement DD2, which is in the same job step.

```
//DD4    DD     DSNAME=JST,DISP=(NEW,KEEP),UNIT=3350,
//             SPACE=(CYL,(12,2)),DCB=(A.B.C,KEYLEN=8)
```

DD statement DD4 defines a new data set named JST and requests that the system copy the DCB information from the data set label of the cataloged data set named A.B.C. If the data set label contains a key length specification, it is overridden by the KEYLEN coded on this DD statement.

```
//DD5    DD     DSNAME=SMAE,DISP=OLD,
//             DCB=(*.STEP1.PROCSTP5.DD8,BUFNO=5)
```

DD statement DD5 defines an existing, cataloged data set named SMAE and requests that the system copy DCB subparameters from DD statement DD8, which is contained in the procedure step named PROCSTP5. The cataloged procedure is called by EXEC statement STEP1. Any of the DCB subparameters coded on DD statement DD8 are ignored if they are specified in the program. If the DCB BUFNO subparameter is not specified in the program, five buffers are assigned.

DCB Subparameters

DCB Subparameters	Access Method: BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
BFALN	X	X	X	X		X		X	X		BFALN = {F|D} Specifies that each buffer starts either on a word boundary that is not also a doubleword boundary or on a doubleword boundary. If both BFALN and BFTEK are specified, they must be specified from the same source. Default: D (doubleword) Note: Do not code the BFALN subparameter with DCB subparameter GNCP, or DD parameters DDNAME, QNAME.
BFTEK	X			X	X				X		BFTEK = R for BDAM and BSAM BFTEK = D for BTAM BFTEK = {S|E|A} for QSAM R Specifies that the data set is being created for or contains variable-length spanned records. D Specifies that dynamic buffering is to be used in the processing program; if dynamic buffering is specified, a buffer pool must also be defined. S, E, and A Specify simple, exchange, or locate mode logical record interface for spanned records. S, E, or A can be coded only when RECFM = VS. If both BFALN and BFTEK are specified, they must be specified from the same source. Note: Do not code the BFTEK subparameter with DCB subparameter GNCP, or DD parameters DDNAME, QNAME.
BLKSIZE	X		X	X		X		X	X	X	BLKSIZE = bytes Specifies the maximum length, in bytes, of a block. The maximum is 32760. The number you specify for BLKSIZE depends on the device type and the record format for the data set. For ASCII data sets on magnetic tape, the minimum value for BLKSIZE is 18 bytes and the maximum is 2048 bytes. If you code the BLKSIZE subparameter in the DCB macro instruction or on a DD statement that defines an existing data set with standard labels, the DCB BLKSIZE overrides the block size specified in the label. BLKSIZE can be coded but will have no effect on EXCP processing. Default: If you do not code BLKSIZE, the system determines an optimum block size for the data. Note: Do not code the BLKSIZE subparameter with DCB subparameter BUFSIZE.
BUFIN										X	BUFIN = buffers Specifies the number of buffers to be assigned initially for receiving operations for each line in the line group. The combined BUFIN and BUFOUT values must not be greater than the number of buffers in the buffer pool for this line group (not including those for disk activity only). Default: 1 Note: Do not code the BUFIN subparameter with DCB subparameter BUFNO, or DD parameters DDNAME, QNAME.
BUFL	X	X	X	X		X		X	X	X	BUFL = bytes Specifies the length, in bytes, of each buffer in the buffer pool. The maximum is 32760. Note: Do not code the BUFL subparameter with DD parameter DDNAME.

DCB Subparameters	Access Method: BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
BUFMAX										X	BUFMAX = buffers Specifies the maximum number of buffers to be allocated to a line at one time. Number must be 2 through 15 and must be equal to or greater than the larger of the numbers specified by the BUFIN and BUFOUT subparameters. Default: 2 Note: Do not code the BUFMAX subparameter with DCB subparameter NCP, or DD parameters DDNAME, QNAME.
BUFNO	X	X	X	X	X	X		X	X		BUFNO = buffers Specifies the number of buffers to be assigned to the DCB. The maximum normally is 255, but can be less because of the size of the region. Note: Do not code the BUFNO subparameter with DCB subparameters BUFIN, BUFOUT, or DD parameter QNAME.
BUFOFF				X					X		BUFOFF = {n|L} n Specifies the length, in bytes, of the block prefix used with an ASCII tape data set. For input, n can be 0 through 99. For output, n must be 0 for writing an output data set with fixed-length or undefined-length records. L Specifies that the block prefix is 4 bytes and contains the block length. BUFOFF = L is valid only with RECFM = D. For output, only BUFOFF = L is valid. Note: Do not code the BUFOFF subparameter with DD parameters DDNAME, QNAME.
BUFOUT										X	BUFOUT = buffers Specifies the number of buffers to be assigned initially for sending operations for each line in the line group. The combined number of BUFIN and BUFOUT values must not be greater than the number of buffers in the buffer pool for this line group (not including those for disk activity only) and cannot exceed 15. Default: 2 Note: Do not code the BUFOUT subparameter with DCB subparameter BUFNO, or DD parameter DDNAME.
BUFSIZE										X	BUFSIZE = bytes Specifies the length, in bytes, of each of the buffers to be used for all lines in a particular line group. Length must be 31 through 65535 bytes. Note: Do not code the BUFSIZE subparameter with DCB subparameter BLKSIZE, or DD parameters DDNAME, QNAME.
CPRI										X	CPRI = {R|E|S} Specifies the relative transmission priority assigned to the lines in this line group. R Specifies that processor receiving has priority over processor sending. E Specifies that receiving and sending have equal priority. S Specifies that processor sending has priority over processor receiving. Note: Do not code the CPRI subparameter with DCB subparameter THRESH, or DD parameters DDNAME, OUTLIM, QNAME.
CYLOFL								X			CYLOFL = tracks Specifies the number of tracks on each cylinder to hold the records that overflow from other tracks on that cylinder. The maximum is 99. Specify CYLOFL only when OPTCD = Y. Note: Do not code the CYLOFL subparameter with DCB subparameter RESERVE, or DD parameters DDNAME, FCB, QNAME, UCS.

DCB Subparameters	Access Method: BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
DEN				X		X			X		DEN = {1|2|3|4} Specifies the magnetic density, in number of bytes-per-inch, used to write a magnetic tape data set. DEN: 1 — 7-track tape: 556 — 9-track tape: - DEN: 2 — 7-track tape: 800 — 9-track tape: 800 (NRZI) DEN: 3 — 7-track tape: - — 9-track tape: 1600 (PE) DEN: 4 — 7-track tape: - — 9-track tape: 6250 (GCR) NRZI Non-return-to-zero inverted recording mode. PE Phase encoded recording mode. GCR Group coded recording mode. Default: 800 bpi assumed for 7-track tape and 9-track without dual density. 1600 bpi assumed for 9-track with dual density or phase-encoded drives. 6250 bpi assumed for 9-track with 6250/1600 bpi dual density or group coded recording tape. Note: Do not code the DEN subparameter with DD parameters DDNAME, QNAME.
DIAGNS	X	X	X	X	X	X	X	X	X		DIAGNS = TRACE Specifies the OPEN/CLOSE/EOV trace option, which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB. If the generalized trace facility (GTF) is not running and tracing user events, DIAGNS is ignored.
DSORG	X	X	X	X	X	X	X	X	X	X	DSORG = organization Specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable. Note: Do not code the DSORG subparameter with DD parameters DDNAME, QNAME, RECORG.

	Organization	Access Method
PS	Physical sequential data set	BSAM,EXCP,QSAM,TCAM
PSU	Physical sequential data set that contains location-dependent information	BSAM,QSAM,EXCP
DA	Direct access data set	BDAM,EXCP
DAU	Direct access data set that contains location-dependent information	BDAM,EXCP
IS	Indexed sequential data set	BISAM,QISAM,EXCP
ISU	Indexed sequential data set that contains location-dependent information	QISAM,EXCP
PO	Partitioned data set	BPAM,EXCP
POU	Partitioned data set that contains location-dependent information	BPAM,EXCP
CX	Communications line group	BTAM
GS	Graphic data control block	GAM

DCB Subparameters	Access Method: BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
EROPT					X				X		EROPT=x BTAM: Requests the BTAM on-line terminal test option. x=T QSAM: Specifies the option to be executed if an error occurs in reading or writing a record. x=ACC System is to accept the block causing the error. SKP System is to skip the block causing the error. ABE System is to cause abnormal end of task. Default: ABE Note: Do not code the EROPT subparameter with DD parameters DDNAME, QNAME.
FUNC				X					X		FUNC={I|R|P|W|D|X|T} Specifies the type of data set to be opened for a 3505 Card Reader or 3525 Card Punch. Unpredictable results will occur if coded for other than a 3505 or 3525. I Data set is for punching and printing cards. R Data set is for reading cards. P Data set is for punching cards. W Data set is for printing. D Protected data set is for punching. X Data set is for both punching and printing. T Two-line print option. The only valid combinations of these values are: I WT RWT RPWXT PWX R RP PW RPWD RPWX P RPD PWXT RWX RWX W RW RPW RWXT Default: P, for output data set. R, for input data set. Note: Do not code the FUNC subparameter with the data-set-sequence number of the DD LABEL parameter, or DD parameters DDNAME, QNAME.
GNCP							X				GNCP=n Specifies the maximum number of I/O macro instructions that the program will issue before a WAIT macro instruction. Note: Do not code the GNCP subparameter with DCB subparameters BFALN, BFTEK, or DD parameters DDNAME, QNAME.
INTVL										X	INTVL={n|0} Specifies the interval, in seconds, between passes through an invitation list. Default: 0 Note: Do not code the INTVL subparameter with DD parameters DDNAME, FCB, QNAME, UCS.
IPLTXID										X	IPLTXID=member Specifies the name of the partitioned data set member that you want loaded into a 3704/3705 Communications Controller. The DCB IPLTXID subparameter overrides IPLTXID in the TERMINAL macro representing the NCP. Note: Do not code the IPLTXID subparameter with DD parameters DDNAME, DSNAME, QNAME.

DCB Subparameters	Access Method: BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
KEYLEN	X		X	X		X		X		X	KEYLEN = bytes Information for the DCB KEYLEN keyword subparameter is moved to the DD KEYLEN parameter described on page 10-104.
LIMCT	X										LIMCT = {blocks|tracks} Specifies how many blocks (if relative block addressing is used) or how many tracks (if relative track addressing is used) are to be searched for a free block or available space. This kind of search occurs only when DCB OPTCD = E is also specified; otherwise, LIMCT is ignored. If the LIMCT number equals or exceeds the number of blocks or tracks in the data set, the entire data set is searched. Note: Do not code the LIMCT subparameter with DD parameters DDNAME, QNAME.
LRECL			X	X		X		X	X	X	LRECL = bytes Information for the DCB LRECL keyword subparameter is moved to the DD LRECL parameter described on page 10-117.
MODE				X		X			X		MODE = $\left\{ \begin{matrix} C \\ E \end{matrix} \left[\begin{matrix} O \\ R \end{matrix} \right] \right\}$ Specifies the mode of operation to be used with a card reader, a card punch, or a card read-punch. C Card image (column binary) mode E EBCDIC mode O Optional mark read mode R Read column eliminate mode If you specify R, you must also specify either C or E. Do not code the MODE subparameter for data entered through the input stream except in a JES3 system. Do not code MODE = C for JES2 or JES3 output. Default: E Note: Do not code the MODE subparameter with DCB subparameters KEYLEN, PRTSP, TRTCH, or DD parameters DDNAME, KEYLEN, QNAME.
NCP		X	X	X							NCP = n Specifies the maximum number of READ or WRITE macro instructions that will be issued before a CHECK macro instruction is issued to test for completion of the I/O operation. The maximum number is 99, but may actually be smaller depending on the size of the region or partition. If chained scheduling is used, the number must be greater than 1. Default: 1 Note: Do not code the NCP subparameter with DCB subparameter BUFMAX, or DD parameters DDNAME, QNAME.

DCB Subparameters	Access Method: BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
NTM								X			NTM = tracks Specifies the number of tracks to be used for a cylinder index. When the specified number of tracks has been filled, a master index is created. The DCB NTM is needed only when the DCB OPTCB = M. If you specify OPTCD = M but omit NTM, the master index option is ignored. Note: Do not code the NTM subparameter with DCB subparameter PCI, or DD parameters DDNAME, QNAME.
OPTCD	X	X	X	X		X		X	X	X	Specifies the optional services to be performed by the control program. All optional services must be requested in one source, that is, in the data set label of an existing data set, in the DCB macro, or in the DD DCB parameter. However, the processing program can modify the DCB OPTCD field. Code the characters in any order; when coding more than one, do not code commas between the characters. Note: Do not code the OPTCD subparameter with DD parameter DDNAME. BDAM: OPTCD = {A|R} [E][F][W] A indicates that the actual device addresses are to be specified in READ and WRITE macro instructions. R indicates that relative block addresses are to be specified in READ and WRITE macro instructions. E indicates that an extended search (more than one track) is to be performed for a block of available space. LIMCT must also be coded. Do not code LIMCT = 0 because it will cause an abnormal termination when a READ or WRITE macro instruction is executed. F indicates that feedback can be requested in READ and WRITE macro instructions and the device is to be identified in the same form as it was presented to the control program. W requests a validity check for write operations on direct access devices and the IBM 3480 Magnetic Tape Subsystem. BISAM: OPTCD = {[L][R][W]} L requests that the control program delete records that have a first byte of all ones. These records will be deleted when space is required for new records. To use the delete option, the DCB RKP must be greater than zero for fixed-length records and greater than four for variable-length records. R requests that the control program place reorganization criteria information in certain fields of the DCB. The problem program can analyze these statistics to determine when to reorganize the data set. W requests a validity check for write operations on direct access devices. Default: R, whenever the OPTCD subparameter is omitted from all sources. BPAM: OPTCD = {C|W|CW / C|H|HC / C|WH|WHC} C requests chained scheduling. W requests a validity check for write operations on direct access devices. H requests that a partitioned data set being processed and residing on MSS, if opening for input, is to be staged to end of file (EOF) on the virtual DASD. Otherwise, only the directory is staged.

DCB Subparameters	Access Method: BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
OPTCD (continued)											BSAM and QSAM: OPTCD = { B | T | U[C] | C[T][B][U] | H[Z][B] | J[C][U] | W[C][T][B][U] | Z[C][T][B][U] | Q[C][T][B] | Z } B requests that the end-of-volume (EOV) routine disregard the end-of-file (EOF) recognition for magnetic tape. For an input data set on a standard-labeled (SL or AL) tape, the EOV routine treats EOF labels as EOV labels until the volume serial list is exhausted. This option allows SL or AL tapes to be read out of volume sequence or to be concatenated to another tape with the same data set name using one DD statement. C requests chained scheduling. H requests hopper empty exit for optical readers or bypass of DOS checkpoint records. J for a data set to be printed on a 3800 Printing Subsystem, instructs the system that the logical record for each output data line contains a table reference character (TRC). The TRC identifies which character arrangement table in the CHARS parameter is to be used to print the line. Before specifying OPTCD = J, see the *3800 Printing Subsystem Programmer's Guide*. Q requests (1) that ASCII tape records in an input data set be converted to EBCDIC code when the input record has been read, or (2) an output record in EBCDIC code be converted to ASCII code before the record is written. T requests user totaling facility. T cannot be specified for a SYSIN or sysout data set. U for 1403 or 3211 Printers with the Universal Character Set (UCS) feature and for the 3800, permits data checks and allows analysis by an appropriate error analysis routine. If U is omitted, data checks are not recognized as errors. U for MSS, requests window processing to reduce the amount of staging space required to process large sequential data sets. The DCB DSORG must be PS, the allocation must be in cylinders, and the type of I/O accessing must be INPUT only or OUTPUT only. W requests a validity check for write operations on direct access devices and the IBM 3480 Magnetic Tape Subsystem. Z for magnetic tape input, requests that the control program shorten its normal error recovery procedure. When specified, a data check is considered permanent after five unsuccessful attempts to read a record. OPTCD = Z is ignored if chained scheduling is used.
											EXCP: OPTCD = Z Z for magnetic tape input, requests that the control program shorten its normal error recovery procedure. When specified, a data check is considered permanent after five unsuccessful attempts to read a record.

DCB Subparameters	Access Method: BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
OPTCD (continued)											QISAM: OPTCD = {[I][L][M][R][U][W][Y]} I requests that ISAM use the independent overflow areas for overflow records. L requests that ISAM delete records that have a first byte of all ones. These records can be deleted when space is required for new records. To use the delete option, the DCB RKP must be greater than zero for fixed-length records and greater than four for variable-length records. M requests that the system create and maintain one or more master indexes, according to the number of tracks specified in the DCB NTM subparameter. R requests that the control program place reorganization criteria information in the DCB. The problem program can analyze these statistics to determine when to reorganize the data set. U requests that the system accumulate track index entries in storage and write them as a group for each track of the track index. U can be specified only for fixed-length records. W requests a validity check for write operations on direct access devices. Y requests that the system use the cylinder overflow areas for overflow records. Default: R, whenever the OPTCD subparameter is omitted from all sources.
											TCAM: OPTCD = {C|U|W} C specifies that one byte of the work area indicates if a segment of a message is the first, middle, or last segment. U specifies that the work unit is a message. If U is omitted, the work unit is assumed to be a record. W specifies that the name of each message source is to be placed in an 8-byte field in the work area.
PCI										X	PCI = {([N][,N]) / ([R][,R]) / ([A][,A]) / ([X][,X])} Specifies (1) whether or not a program-controlled interruption (PCI) is to be used to control the allocation and freeing of buffers and (2) how these operations are to be performed. The first operand applies to receiving operations and the second to sending operations. N specifies that no PCIs are taken while filling buffers during receiving operations or emptying buffers during sending operations. R specifies that after the first buffer is filled or emptied, a PCI occurs during the filling or emptying of each succeeding buffer. The completed buffer is freed, but no new buffer is allocated to take its place. A specifies that after the first buffer is filled or emptied, a PCI occurs during the filling or emptying of the next buffer. The first buffer is freed, and a buffer is allocated to take its place. X specifies that after a buffer is filled or emptied, a PCI occurs during the filling or emptying of the next buffer. The first buffer is not freed, but a new buffer is allocated. You can omit the parentheses if you code only the first operand. Default: (A,A) Note: Do not code the PCI subparameter with DCB subparameter NTM, or DD parameters DDNAME, QNAME.

DCB Subparameters	Access Method: BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
PRTSP				X		X			X		PRTSP={0|1|2|3} Specifies the line spacing for an online printer. PRTSP is valid only for an online printer and only if the RECFM is not A or M. PRTSP=2 is ignored if specified with the DD SYSOUT parameter. 0 - spacing is suppressed, 1 - single, 2 - double, 3 - triple spacing JES2 ignores PRTSP for sysout data sets. Default: 1 Note: Do not code the PRTSP subparameter with DCB subparameters KEYLEN, MODE, STACK, TRTCH, or DD parameters DDNAME, KEYLEN, QNAME.
RECFM	X		X	X		X		X	X	X	RECFM=format Information for the DCB RECFM keyword subparameter is moved to the DD RECFM parameter described on page 10-136.
RESERVE										X	RESERVE=(bytes1,bytes2) Specifies the number of bytes (0 through 255) to be reserved in a buffer for insertion of data by the DATETIME and SEQUENCE macros. bytes1 indicates the number of bytes to be reserved in the first buffer that receives an incoming message. bytes2 indicates the number of bytes to be reserved in all the buffers following the first buffer in a multiple-buffer header situation. Default: (0,0) Note: Do not code the RESERVE subparameter with DCB subparameters CYLOFL, RKP, or DD parameters DDNAME, KEYOFF, QNAME, UCS.
RKP						X		X			RKP=number With SMS, use the DD KEYOFF or DATACLAS parameter. Specifies the position of the first byte of the record key in each logical record. The first byte of a logical record is position 0. If RKP=0 is specified for blocked fixed-length records, the key begins in the first byte of each record. OPTCD=L must not be specified. If RKP=0 is specified for unblocked fixed-length records, the key is not written in the data field. OPTCD=L can be specified. For variable-length records, the relative key position must be 4 or greater, if OPTCD=L is not specified; the relative key position must be 5 or greater, if OPTCD=L is specified. For EXCP processing, RKP can be coded but is ignored. Default: 0 Note: Do not code the RKP subparameter with DCB subparameter RESERVE, or DD parameters DDNAME, FCB, KEYOFF, UCS.

DCB Subparameters	Access Method										Description of Subparameters
	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	
STACK				X		X			X		STACK={1|2} Specifies which stacker bin is to receive a card. Default: 1 Note: Do not code the STACK subparameter with DCB subparameters KEYLEN, PRTSP, TRTCH, or DD parameters DDNAME, KEYLEN, QNAME.
THRESH										X	THRESH=nn Specifies the percentage of the nonreusable disk message queue records that are to be used before a flush closedown occurs. Default: Closedown occurs when 95 percent of the records have been used. Note: Do not code the THRESH subparameter with DCB subparameter CPRI, or DD parameters DDNAME, OUTLIM, QNAME.
TRTCH				X		X			X		TRTCH={C|E|T|ET} Specifies the recording technique for 7-track tape. C specifies data conversion, odd parity, and no translation. E specifies no data conversion, even parity, and no translation. T specifies no data conversion, odd parity, and that BCD to EBCDIC translation is required when reading and EBCDIC to BCD translation when writing. ET specifies no data conversion, even parity, and that BCD to EBCDIC translation is required when reading and EBCDIC to BCD translation when writing. Default: no conversion, odd parity, and no translation. Note: Do not code the TRTCH subparameter with DCB subparameters KEYLEN, MODE, PRTSP, STACK, or DD parameters DDNAME, KEYLEN, QNAME.

DDNAME Parameter

Parameter Type: Keyword, optional

Purpose: Use the DDNAME parameter to postpone defining a data set until later in the same job step. A DDNAME parameter on a DD statement in a cataloged or in-stream procedure allows you to postpone defining the data set until a job step calls the procedure; the data set must be defined in the calling job step.

Syntax:

```
DDNAME=ddname
```

Subparameter Definition

ddname
Refers to a later DD statement that defines the data set. **ddname** must match the ddname of the referenced DD statement.

A job step or procedure step can contain up to five DD statements with DDNAME parameters. Each DDNAME parameter must refer to a different DD statement.

Overrides

If any DCB subparameter appears on both DD statements, the DCB subparameter on the referenced DD statement overrides the DCB subparameter on the DD statement that contains DDNAME.

Relationship to Other Parameters

The **only** DD parameters you can code with the DDNAME parameter are:

```
DCB=BLKSIZE    LIKE
DCB=BUFNO      REFDD
DCB=DIAGNS
```

Do not code the DDNAME parameter on a DD statement with a ddname of JOBLIB, JOBCAT, or STEPCAT.

Location in the JCL

Place a DD statement containing a DDNAME parameter in a job step or in a cataloged or in-stream procedure. The referenced DD statement must be later in the same job step, must be in the calling job step, or must be in a cataloged or in-stream procedure called by the job step.

Location of DD Statements for Concatenated Data Sets: 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 referenced DD statement that defines the data set.

Errors in Location of Referenced DD Statement: The system treats a DDNAME parameter as through it were a DUMMY parameter and issues a warning message in the following cases:

- If the job step or called procedure does not contain the referenced DD statement.
- If the referenced DD statement appears earlier in the job step.

Location of DD Statement Requesting Unit Affinity: To use the same device, a DD statement can request unit affinity to an earlier DD statement by specifying UNIT=AFF=ddname. If a DD statement requests unit affinity to a DD statement containing a DDNAME parameter, the DD statement requesting unit affinity must be placed after the referenced DD statement. If the DD statement requesting unit affinity appears before, the system treats the DD statement requesting unit affinity as a DUMMY DD statement.

```
//STEP   EXEC   PGM=TKM
//DD1    DD     DDNAME=DD4
//DD2    DD     DSNAME=A,DISP=OLD
          .
          .
//DD4    DD     DSNAME=B,DISP=OLD
//DD5    DD     UNIT=AFF=DD1
```

DD1 postpones defining the data set until DD4. DD5 requests unit affinity to DD1. Because DD1 has been defined when DD5 is processed, the system assigns DD5 to the same device as DD1.

Instead of specifying UNIT=AFF=ddname, both DD statements can specify the same devices in their UNIT parameters or the same volume serials in their VOLUME parameters.

Referenced DD Statement

If the DDNAME parameter appears in a procedure with multiple steps, the ddname on the referenced DD statement takes the form stepname.ddname. For example, if procedure step STEPCP1 contains:

```
//INDATA   DD   DDNAME=DD1
```

The referenced DD statement in the calling job step is:

```
//STEPCP1.DD1   DD *
```

Parameters not Permitted on the Referenced DD Statement: The referenced DD statement must not contain a DYNAM parameter.

A DD statement that contains a DDNAME parameter must not override a procedure sysout DD statement that contains an OUTPUT parameter if the referenced DD statement also contains an OUTPUT parameter.

Backward References

A backward reference is a reference to an earlier DD statement in the job or in a cataloged or in-stream procedure called by a job step. A backward reference is in the form *.ddname or *.stepname.ddname or *.stepname.procstepname.ddname. The ddname in the reference is the ddname of the earlier DD statement. If the earlier DD statement contains a DDNAME parameter, the reference is to the ddname in the name field of the earlier statement, **not** to the ddname in the DDNAME parameter.

The DD statement referenced in a DDNAME parameter cannot refer to a DD statement between the statement containing the DDNAME parameter and itself. For example:

```
//SHOW    EXEC PGM=ABLE
//DD1     DD   DDNAME=INPUT
//DD2     DD   DSNAME=TEMPSPAC,SPACE=(TRK,1),UNIT=SYSDA
//DD3     DD   DSNAME=INCOPY,VOLUME=REF=*.DD1,
//             DISP=(,KEEP),SPACE=(TRK,(5,2))
//DD4     DD   DSNAME=OUTLIST,DISP=OLD
//DD5     DD   DSNAME=MESSAGE,DISP=OLD,UNIT=3330,VOLUME=SER=333333
//INPUT   DD   DSNAME=NEWLIST,DISP=(OLD,KEEP),VOLUME=SER=333333,
//             UNIT=3330
```

The DDNAME parameter on DD1 refers to DD statement INPUT. The VOLUME parameter of DD3 specifies a backward reference to DD1, which is the name field ddname.

DD statement INPUT identifies the volume 333333 in its VOLUME=SER=333333 parameter. DD statement INPUT cannot use a backward reference to the VOLUME parameter on DD5 because DD5 is between the referring DD1 and the referenced INPUT.

Examples of the DDNAME Parameter

The following procedure step is the only step in a cataloged procedure named CROWE:

```
//PROCSTEP  EXEC PGM=RECPGM
//DD1       DD   DDNAME=WKREC
//POD       DD   DSNAME=OLDREC,DISP=OLD
```

DD statement DD1 is intended for weekly records in the input stream; these records are processed by this step. Because the * and DATA parameters cannot be used in cataloged procedures, the DDNAME parameter is coded to postpone defining the data set until the procedure is called by a job step. The step that calls the procedure is:

```
//STEPA  EXEC PROC=CROWE
//WKREC  DD   *
         .
         .
         data
         .
/*
```

When the procedure contains multiple steps, use the form stepname.ddname for the ddname of the referenced DD statement. For example, the following procedure steps appear in a cataloged procedure named PRICE:

```
//STEP1  EXEC PGM=SUGAR
//DD1    DD   DDNAME=QUOTES
          .
          .
          .
//STEP2  EXEC PGM=MOLASS
//DD2    DD   DSNAME=WEEKB,DISP=OLD
          .
          .
          .
```

The step that calls the procedure is:

```
//STEPA         EXEC PROC=PRICE
//STEP1.QUOTES DD   *
                .
                .
                data
                .
/*
```

When the referenced DD statement is to be a concatenation, the procedure must already contain the concatenation. (Such as when the referencing DD statement is to contain instream data.) For example, the following procedure step appears in cataloged procedure NEWONE.

```
//NEWONE PROC
//STEP1  EXEC  PGM=TRYIT
//DD1    DD    DDNAME=INSTUFF
//       DD    DSN=OLDSTUFF,DISP=OLD
          .
          .
```

The step that calls the procedure is:

```
//STEPA  EXEC  PROC=NEWONE
//STEP1.INSTUFF DD  *
          .
         data
          .
/*
```

The instream data (DDNAME=INSTUFF) is inserted before OLDSTUFF in the concatenation.

DEST Parameter

Parameter Type: Keyword, optional

Purpose: Use the DEST parameter to specify a destination for a sysout data set. The DEST parameter can send a sysout data set to a remote or local terminal, a node, a node and remote work station, a local device or group of devices, or a node and userid.

Note: Code the DEST parameter only on a DD statement with a SYSOUT parameter. Otherwise, the system checks the DEST parameter for syntax, then ignores it.

Syntax:

```
DEST=destination

The destination subparameter for JES2 is one of the following:

     LOCAL
     name
     Nnnnn
     NnnRmmmm
     NnnnRmmm
     NnnnnRmm
     Rnnnn
     RMnnnn
     RMTnnnn
     Unnnn
     (node,userid)

The destination subparameter for JES3 is one of the following:

     ANYLOCAL
     device-name
     device-number
     group-name
     nodename
     (node,userid)
```

Subparameter Definition for JES2 Systems

LOCAL
Indicates any local device.

name
Identifies a local or remote device by a symbolic name defined by the installation during JES2 initialization. The name is 1 through 8 alphanumeric or national ($, #, @) characters.

Nnnnn
Identifies a node. nnnn is 1 through 4 decimal numbers from 1 through 1000.

NnnRmmmm
NnnnRmmm
NnnnnRmm
Identifies a node and a remote work station connected to the node. The node number, indicated in the format by n, is 1 through 4 decimal numbers from 1 through 1000. The remote work station number, indicated in the format by m, is 1 through 4 decimal numbers from 1 through 9999. Do not code leading zeros in n or m. The maximum number of digits for n and m combined cannot exceed six.

Note: R0 is equivalent to specifying LOCAL at node Nn.

Rnnnn
RMnnnn
RMTnnnn
Identifies a remote terminal. nnnn is 1 through 4 decimal numbers from 1 through 9999. Note that with remote pooling, the installation may translate this route code to another route code.

Note: R0 is equivalent to LOCAL.

Unnnn
Identifies a local terminal with special routing. nnnn is 1 through 4 decimal numbers from 1 through 9999.

(node,userid)
Identifies a node and a TSO or VM userid at that node. The node is a symbolic name defined by the installation during initialization; node is 1 through 8 alphanumeric or national ($, #, @) characters. The userid must be defined at the node; userid for TSO is 1 through 7 alphanumeric or national ($, #, @) characters and for VM is 1 through 8 alphanumeric or national ($, #, @) characters.

A userid requires a node; therefore, code DEST=(node,userid). You **cannot** code a userid without a node.

DEST=(node) is valid with a **writer-name** subparameter in the SYSOUT parameter; however, DEST=(node,userid) is invalid. Therefore, you can code SYSOUT=(A,writer-name),DEST=(node).

Note: If a data set is queued for transmission and an operator changes its destination, the userid portion of the routing is lost.

Subparameter Definition for JES3 Systems

ANYLOCAL
Indicates any local device that is attached to the global processor.

device-name
Identifies a local device by a symbolic name defined by the installation during JES3 initialization. device-name is 1 through 8 alphanumeric or national ($, #, @) characters.

device-number
Identifies the 3-character device number.

group-name
Identifies a group of local devices, an individual remote station, or a group of remote stations by a symbolic name defined by the installation during JES3 initialization. group-name is 1 through 8 alphanumeric or national ($, #, @) characters.

nodename
Identifies a node by a symbolic name defined by the installation during JES3 initialization. nodename is 1 through 8 alphanumeric or national ($, #, @) characters. If the nodename you specify is the same as the node you are working on, JES3 treats the output as though you specified ANYLOCAL.

(node,userid)
Identifies a node and a TSO or VM userid at that node. The node is a symbolic name defined by the installation during initialization; node is 1 through 8 alphanumeric or national ($, #, @) characters. The userid must be defined at the node; userid for TSO is 1 through 7 alphanumeric or national ($, #, @) characters and for VM is 1 through 8 alphanumeric or national ($, #, @) characters.

A userid requires a node; therefore, code DEST=(node,userid). You **cannot** code a userid without a node.

DEST=(node) is valid with a **writer-name** subparameter in the SYSOUT parameter: however, DEST=(node,userid) is invalid. Therefore, you can code SYSOUT=(A,writer-name),DEST=(node).

Defaults

If you do not code a DEST parameter, JES directs the sysout data set to the default destination for the input device from which the job was submitted.

If a specified destination is invalid, the job fails.

Overrides

The DEST parameter on the sysout DD statement overrides an OUTPUT JCL DEST parameter.

Relationship to Other Parameters

Code the DEST parameter only on a DD statement with the SYSOUT parameter.

Relationship to Other Control Statements

You can also code an output destination using:

- The OUTPUT JCL statement.
- The JES2 /*OUTPUT and /*ROUTE control statements.
- The JES3 //*MAIN, //*FORMAT PR, and //*FORMAT PU control statements.

Because DEST=(node,userid) cannot be coded on JES2 or JES3 control statements, you must code it, if needed, on a DD or OUTPUT JCL statement.

Examples of the DEST Parameter

```
//JOB01   JOB  ,'MAE BIRD',MSGCLASS=B
//STEP1   EXEC PGM=INTEREST
//DEBIT   DD   SYSOUT=A
//CALIF   DD   SYSOUT=A,DEST=R555
//FLOR    DD   SYSOUT=A,DEST=(BOCA,'9212U28')
```

In this example, the system sends the sysout data set defined by DD statement DEBIT to the work station that submitted the job, the data set defined by DD statement CALIF to the remote terminal 555, and the data set defined by DD statement FLOR to VM userid 9212U28 at node BOCA.

DISP Parameter

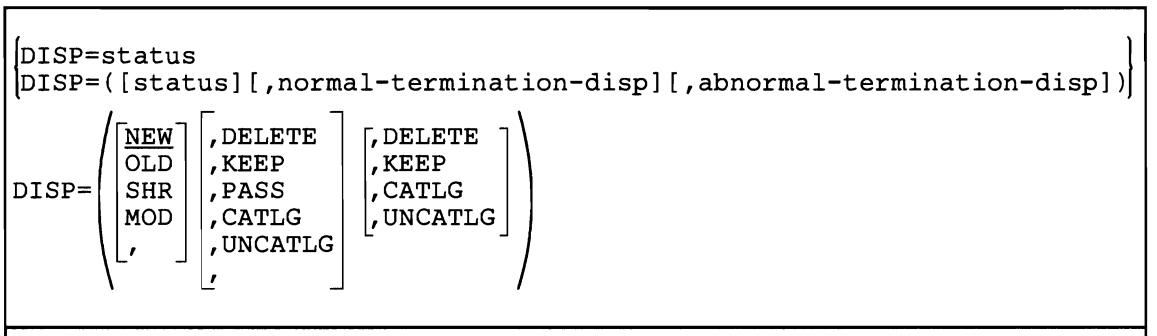
Parameter Type: Keyword, optional

Purpose: Use the DISP parameter to describe the status of a data set to the system and tell the system what to do with the data set after termination of the step or job. You can specify one disposition for normal termination and another for abnormal termination.

Note: Disposition of the **data set** is controlled solely by the DISP parameter; disposition of the **volume(s)** on which the data set resides is a function of the volume status when the volume is demounted.

References: For information about tape data set processing, see *Magnetic Tape Labels and File Structure Administration.*

Syntax:



- You can omit the parentheses if you code only the status subparameter.
- If you omit the status subparameter but code subparameters for normal or abnormal termination disposition, you must code a comma to indicate the absence of NEW. For example, DISP=(,KEEP) or DISP=(,CATLG,DELETE).
- If you omit the second subparameter but code the third, you must code a comma to indicate the absence of the second subparameter. For example, DISP=(OLD,,DELETE) or DISP=(,,KEEP).

Subparameter Definition

Status Subparameter

NEW
Indicates that a new data set is to be created in this step.

OLD
Indicates that the data set exists before this step and that this step requires exclusive (unshared) use of the data set.

If you specify DISP=OLD for an output tape data set and (1) the data set is not protected by RACF or a password or (2) the data set has no expiration date, the system does not verify the data set name in the header label.

SHR

Indicates that the data set exists before this step and that another job can share it, that is, use it at the same time. This subparameter can also be coded as SHARE.

If you specify DISP=SHR for an output tape data set and (1) the data set is not protected by RACF or a password or (2) the data set has no expiration date, the system does not verify the data set name in the header label.

MOD

Indicates one of the following:

- The data set exists. Records are to be added to the end of the data set. The data set must be sequential.
- A new data set is to be created.

In either case, MOD specifies exclusive (unshared) use of the data set.

When the data set is opened, the read/write mechanism is positioned after the last sequential record for an existing data set or at the beginning for a new data set. For subsequent OPENs within the same step, the read/write mechanism is positioned after the last sequential record.

Note: You cannot specify DISP=MOD to extend ISO/ANSI/FIPS Version 3 tape data sets, unless the ISO/ANSI/FIPS Version 3 label validation installation exit allows the extension. For information on using ISO/ANSI/FIPS Version 3 installation exits, see *Magnetic Tape Labels and File Structure Administration.*

To use DISP=MOD to create a new data set, specify the following:

- No VOLUME=SER or VOLUME=REF parameter on the DD statement. The data set must not be cataloged or passed from another job step.
- A VOLUME=REF parameter that refers to a DD statement that makes a nonspecific volume request. One of the following must also be true:
  - The DSNAME parameters in the two DD statements must be different.
  - The two DD statements must request different areas of the same ISAM data set.

  A nonspecific volume request is a DD statement for a new data set that can be assigned to any volume or volumes.

  Note: For a new generation of a generation data group (GDG) data set (where (+n) is greater than 0), VOLUME=REF or VOLUME=SER can be coded.

After the system chooses a volume for a new data set, if the system finds another data set with the same name on that volume, the system will try to allocate a different volume. However, SMS-managed data sets require unique data set names. If a new data set is chosen to be SMS-managed and an existing SMS-managed data set has the same data set name, the request is failed.

In a JES3 system, if you code DISP=MOD for a multivolume data set and any of the volumes are JES3-managed, JES3 will not execute the job until all volumes, including scratch volumes being added, are allocated. Such a job will wait on the queue until all volumes are allocated.

Normal Termination Disposition Subparameter

DELETE
Indicates that the data set's space on the volume is to be released if this step terminates normally. The space can be used for other data sets; the data set is not erased from the space.

An existing data set is deleted only if its retention period or expiration date has passed; otherwise the data set is kept. A new data set is deleted at the end of the step even though a retention period or expiration date is also specified. See the DD EXPDT or RETPD parameters.

If the system retrieves volume information from the catalog because the DD statement does not specify VOLUME=SER or VOLUME=REF, then DELETE implies UNCATLG: the system deletes the data set and removes its catalog entry.

KEEP
Indicates that the data set is to be kept on the volume if this step terminates normally.

Without SMS, only KEEP is valid for VSAM data sets. VSAM data sets should not be passed, cataloged, uncataloged, or deleted.

With SMS, all dispositions are valid for VSAM data sets; however, UNCATLG is ignored.

For new SMS-managed data sets, KEEP implies CATLG.

PASS
Indicates that the data set is to be passed for use by a subsequent step in the same job.

With SMS, the system replaces PASS with KEEP for permanent VSAM data sets. When you refer to the data set later in the job, the system obtains data set information from the catalog.

Note: A data set can be passed only within a job.

CATLG
Indicates that, if the step terminates normally, the system is to place an entry pointing to the data set in the system or user catalog. For CVOL catalogs, the system creates any missing index levels. Note that the data set is kept.

An unopened tape data set is cataloged, unless the volume request is nonspecific or unless the data set is allocated to a dual-density tape drive but no density is specified. A nonspecific volume request is a DD statement for a new data set that can be assigned to any volume or volumes.

For information about the rules for cataloged data set names, refer to *Integrated Catalog Administration: Access Method Services Reference* or *VSAM Catalog Administration: Access Method Services Reference*.

UNCATLG
Indicates that, if the step terminates normally, the system is to delete (1) the entry pointing to the data set in the system or user catalog and (2) unneeded indexes, except for the highest level entry. Note that the data set is kept.

With SMS, UNCATLG is ignored for SMS-managed data sets and VSAM data sets (KEEP is implied).

Abnormal Termination (Conditional) Disposition Subparameter

DELETE
Indicates that the data set's space on the volume is to be released if this step terminates abnormally. The space can be used for other data sets; the data set is not erased from the space.

An existing data set is deleted only if its retention period or expiration date has passed; otherwise the data set is kept. If the data set was being created when the step abnormally terminates, the data set is deleted even though it has an unexpired retention period or expiration date. See the DD EXPDT or RETPD parameters.

If the system retrieves volume information from the catalog because the DD statement does not specify VOLUME=SER or VOLUME=REF, then DELETE implies UNCATLG: the system deletes the data set and removes its catalog entry.

For a cataloged, passed data set, the user catalog is not updated.

KEEP
Indicates that the data set is to be kept on the volume if this step terminates abnormally.

Without SMS, only KEEP is valid for VSAM data sets. VSAM data sets should not be passed, cataloged, uncataloged, or deleted.

With SMS, all dispositions are valid for VSAM data sets; however, UNCATLG is ignored.

For new SMS-managed data sets, KEEP implies CATLG.

CATLG
Indicates that, if the step terminates abnormally, the system is to place an entry pointing to the data set in the system or user catalog. For CVOL catalogs, the system creates any missing index levels. Note that the data set is kept.

An unopened tape data set is cataloged, unless the volume request is nonspecific or unless the data set is allocated to a dual-density tape drive but no density is specified.

For a cataloged, passed data set, the user catalog is not updated. A passed, not received data set is not cataloged if the data set name has a first-level qualifier of a catalog name or alias.

UNCATLG
Indicates that, if this step terminates abnormally, the system is to delete (1) the entry pointing to the data set in the system or user catalog and (2) unneeded indexes, except for the highest level entry. Note that the data set is kept.

For a cataloged, passed data set, the user catalog is not updated.

With SMS, UNCATLG is ignored for SMS-managed data sets and VSAM data sets (KEEP is implied).

Defaults

- If you omit the status subparameter, the default is NEW.
- If you omit the normal termination disposition subparameter, the default is DELETE for a NEW data set or KEEP for an existing data set.
- If you omit the abnormal termination disposition subparameter, the default is the disposition specified or implied by the second subparameter. However, if the second subparameter specified PASS, the default abnormal termination disposition is DELETE for a NEW data set or KEEP for an existing data set.
- If you omit the DISP parameter, the default is a NEW data set with a disposition of DELETE for both normal and abnormal termination disposition. Thus, you can omit the DISP parameter for a data set that is created and deleted during a step.

Relationship to Other Parameters

Do not code the following parameters with the DISP parameter.

*	DDNAME	SYSOUT
BURST	DYNAM	
CHARS	FLASH	
COPIES	MODIFY	
DATA	QNAME	

Disposition of QSAM Data Sets

Do not code DISP=MOD if the data control block (DCB) specifies RECFM=FBS and the data set is processed by QSAM. If you do and a block is shorter than the specified block size, QSAM assumes that the short block is the last block and starts end-of-file processing. By this action, QSAM can embed short blocks in your data set and so affect the number of records per track.

Disposition of Generation Data Sets

See Appendix B in *JCL User's Guide* for additional information about disposition processing for generation data sets.

Disposition of Temporary Data Sets

Specify a normal termination disposition of PASS or DELETE for a temporary data set or for a data set with a system-generated name, that is, when a DSNAME parameter is omitted from the DD statement.

For a temporary data set name, the system ignores any abnormal termination disposition specified in the third subparameter.

Disposition of Partitioned Data Sets

When you specify DISP=MOD or DISP=NEW for a partitioned data set and you also specify a member name in the DSNAME parameter, the member name must not already exist. If the member name already exists, the system terminates the job.

When you specify DISP=OLD for a partitioned data set and you also specify a member name in the DSNAME parameter, the data set must already exist. If the member name already exists and the data set is opened for output, the system replaces the existing member with the new member. If the member name does not already exist and the data set is opened for output, the system adds the member to the data set.

When you specify DISP=MOD for a partitioned data set and you do not specify a member name, the system positions the read/write mechanism at the end of the data set. The system does not make an automatic entry into the directory.

When you specify DISP=MOD for a partitioned data set and you do specify a member name, the system positions the read/write mechanism at the end of the data set. If the member name already exists, the system terminates the job.

DISP=MOD for a Multivolume Data Set

When you code DISP=MOD and the volume information is for a multivolume data set, normally the first volume(s) will be mounted on the devices(s) allocated. Then, if the data set is opened for output, OPEN starts with the last volume. If the number of volumes is more than the number of allocated devices, the system asks the operator to demount the first volume(s) and mount the last. To have the last volume mounted without first mounting and then demounting the first volume(s):

- **For DASD**, code DEFER in the UNIT parameter or a volume sequence number in the VOLUME parameter. If you code VOLUME=REF, you must also code either DEFER in the UNIT parameter or a volume sequence number in the VOLUME parameter.

- **For tape**, code VOLUME=REF or DEFER in the UNIT parameter or a volume sequence number in the VOLUME parameter.

When you code DISP=MOD for a multivolume tape data set, use the volume count and volume sequence number subparameters of the VOLUME parameter to keep the system from positioning the read/write mechanism after the last record on the last volume. For example:

```
//DDEX1   DD   DSNAME=OPER.DATA,DISP=(MOD,KEEP),VOLUME=(,,1,2)
```

The volume sequence number of 1 specifies that you want to use the first volume, and the volume count of 2 specifies that the data set requires two volumes.

If you want to extend a cataloged, multivolume data set and have it properly cataloged after it is kept or passed, code the VOLUME and UNIT parameters to make the system use the values in the system catalog to process the data set. The following DD statement shows how to keep and extend a cataloged multivolume data set using the system catalog. Remember that this data set was created with a volume count of 2.

```
//DDEX2 DD DSNAME=OPER.DATA,DISP=(MOD,KEEP),
//             VOLUME=(,,,3),UNIT=(,P)
```

The VOLUME parameter references the system catalog for volume information about the data set and increases the maximum number of volumes for OPER.DATA. Because the UNIT parameter requests parallel mounting, the system must allocate the same number of units as the number of volumes in the VOLUME parameter; in this case, 3.

The following is an example of the messages in the job log after the job completes.

```
IEF285I     OPER.DATA                                          KEPT
IEF285I     VOL SER NOS= 333001,333002,333003.
IEF285I     OPER.DATA                                          RECATALOGED
IEF285I     VOL SER NOS= 333001,333002,333003.
```

If you do not reference the system catalog when extending cataloged multivolume data sets, the system does not update the system catalog with the newly referenced volumes.

Effect of DCB=dsname Parameter

If the DCB parameter refers to a cataloged data set, the system obtains the volume sequence number from the label of the data set, unless the volume sequence number is coded on the DD statement.

Thus, for the following DD statement, even though DISP=MOD is specified, the system positions the read/write mechanism after the last record on the volume specified in the volume sequence number in the label; this volume may or may not be the last volume.

```
//DD1    DD    DSNAME=MULTI1,DISP=MOD,DCB=CATDD
```

To control which volume is processed, code a volume sequence number.

```
//DD2    DD    DSNAME=MULTI2,DISP=MOD,DCB=CATDD,VOLUME=(,,2)
```

DD: DISP

Summary of Disposition Processing

DISP Subparameters:			Disposition (If Data Set was Allocated):			
	Normal	Abnormal	At Normal	At Abnormal End of Step		At End of Job
Status	Termination Disposition	Termination Disposition	End of Step	Step Abnormally Terminated	If Later Allocation Failed in Step	
NEW permanent data set or MOD treated as new	none	none	deleted	deleted	deleted	
	KEEP	none	kept	kept	deleted	
	DELETE	none	deleted	deleted	deleted	
	CATLG	none	cataloged	cataloged	deleted	
	PASS	none	passed	passed	passed	deleted
	PASS	DELETE KEEP CATLG UNCATLG	passed	passed	passed	If all steps terminated normally: deleted If a step terminated abnormally: third subparameter disposition
	DELETE KEEP CATLG UNCATLG	KEEP	second subparameter disposition	kept	deleted	
	DELETE KEEP CATLG UNCATLG	DELETE	second subparameter disposition	deleted	deleted	
	DELETE KEEP CATLG UNCATLG	CATLG	second subparameter disposition	cataloged	deleted	
NEW temporary data set	none	DELETE KEEP CATLG UNCATLG	deleted	deleted	deleted	
	DELETE	DELETE KEEP CATLG UNCATLG	deleted	deleted	deleted	
	PASS	DELETE KEEP CATLG UNCATLG	passed	deleted	deleted	deleted
NEW data set in step to be automatically restarted	DELETE KEEP PASS CATLG UNCATLG	DELETE KEEP CATLG UNCATLG		deleted		
NEW data set in step to be restarted at checkpoint	DELETE KEEP PASS CATLG UNCATLG	DELETE KEEP CATLG UNCATLG		kept, if being used when checkpoint was taken		

Figure 10-1 (Part 1 of 2). Summary of Disposition Processing

DISP Subparameters:			Disposition (If Data Set was Allocated):			
	Normal	Abnormal	At Normal	At Abnormal End of Step		At End of Job
Status	Termination Disposition	Termination Disposition	End of Step	Step Abnormally Terminated	If Later Allocation Failed in Step	
OLD or SHR or MOD treated as old	none	none	kept	kept	kept	
	KEEP	none	kept	kept	kept	
	DELETE	none	deleted	deleted	kept	
	CATLG	none	cataloged or, if new volumes were added, recataloged	cataloged or, if new volumes were added, recataloged	kept	
	UNCATLG	none	uncataloged	uncataloged	kept	
	PASS	none	passed	passed	passed	kept
	PASS	DELETE KEEP CATLG UNCATLG	passed	passed	passed	If all steps terminated normally: kept, if originally old; deleted, if originally new If a step terminated abnormally: third subparameter disposition
	DELETE KEEP CATLG UNCATLG	KEEP	second parameter disposition	kept	kept, if step was receiving originally old data set;	
	DELETE KEEP CATLG UNCATLG	DELETE	second parameter disposition	deleted	deleted, if step was receiving originally new data set	
	DELETE KEEP CATLG UNCATLG	CATLG	second parameter disposition	cataloged or, if new volumes were added, recataloged		
	DELETE KEEP CATLG UNCATLG	UNCATLG	second parameter disposition	uncataloged		
OLD permanent data set passed to this job step	none	none	kept, if data set was originally new; deleted, if originally old	kept, if data set was originally new; deleted, if originally old		
OLD data set in step to be automatically restarted	DELETE KEEP PASS CATLG UNCATLG	DELETE KEEP CATLG UNCATLG		kept		
OLD data set in step to be restarted at checkpoint	DELETE KEEP PASS CATLG UNCATLG	DELETE KEEP CATLG UNCATLG		kept, if being used when checkpoint was taken		

Figure 10-1 (Part 2 of 2). Summary of Disposition Processing

Examples of the DISP Parameter

```
//DD2   DD  DSNAME=FIX,UNIT=3420-1,VOLUME=SER=44889,
//          DISP=(OLD,,DELETE)
```

DD statement DD2 defines an existing data set and implies by the omitted second subparameter that the data set is to be kept if the step terminates normally. The statement requests that the system delete the data set if the step terminates abnormally.

```
//STEPA  EXEC PGM=FILL
//DD1    DD   DSNAME=SWITCH.LEVEL18.GROUP12,UNIT=3350,
//         VOLUME=SER=LOCAT3,SPACE=(TRK,(80,15)),DISP=(,PASS)
//STEPB  EXEC PGM=CHAR
//DD2    DD   DSNAME=XTRA,DISP=OLD
//DD3    DD   DSNAME=*.STEPA.DD1,DISP=(OLD,PASS,DELETE)
//STEPC  EXEC PGM=TERM
//DD4    DD   DSNAME=*.STEPB.DD3,DISP=(OLD,CATLG,DELETE)
```

DD statement DD1 defines a new data set and requests that the data set be passed. If STEPA abnormally terminates, the data set is deleted because it is a new data set, the second subparameter is PASS, and an abnormal termination disposition is not coded.

DD statement DD3 in STEPB receives this passed data set and requests that the data set be passed. If STEPB abnormally terminates, the data set is deleted because of the third subparameter of DELETE.

DD statement DD4 in STEPC receives the passed data set and requests that the data set be cataloged at the end of the step. If STEPC abnormally terminates, the data set is deleted because of the abnormal termination disposition of DELETE.

DD statement DD2 defines an old data set named XTRA. When STEPB terminates, normally or abnormally, this data set is kept.

```
//SMSDD5 DD  DSNAME=MYDS5.PGM,DATACLAS=DCLAS05,STORCLAS=SCLAS05,
//           DISP=(NEW,KEEP)
```

DD statement SMSDD5 defines a new SMS-managed data set and requests that the data set be kept (which implies that it be cataloged).

```
//SMSDD7 DD  DSNAME=MYDS7.PGM,DISP=(OLD,UNCATLG)
```

DD statement SMSDD7 defines an existing SMS-managed data set (the data set had been assigned a storage class when it was created) and requests that the data set be uncataloged. However, the data set is kept because UNCATLG is ignored for SMS-managed data sets.

DLM Parameter

Parameter Type: Keyword, optional

Purpose: Use the DLM parameter to specify a delimiter to terminate this in-stream data set. When the DLM parameter assigns a different delimiter, the in-stream data records can include standard delimiters, such as /* and //, in the data.

In a JES2 system, when the DLM delimiter appears on a DD * statement, either the assigned delimiter or // ends the input data set. When the DLM delimiter appears on a DD DATA statement, only the assigned delimiter ends the input data set.

In a JES3 system, when the DLM delimiter appears on either a DD * or DD DATA statement, only the assigned delimiter ends the input data set.

Note: When the DLM delimiter overrides any implied delimiter, you must terminate the data with the DLM characters. Otherwise, the system keeps reading until the reader is empty.

Except for the JES2 /*SIGNON and /*SIGNOFF statements, the system does not recognize JES2 and JES3 statements in an input stream between the DLM parameter and the delimiter it assigns. The JES2 /*SIGNON and /*SIGNOFF statements are processed by the remote work station regardless of any DLM delimiter.

Syntax:

```
DLM=delimiter
```

- If the specified delimiter contains any special characters, enclose it in apostrophes. In this case, a special character is any character that is neither alphanumeric nor national ($, #, @).

  Failing to code enclosing apostrophes produces unpredictable results.

- If the delimiter contains an ampersand or an apostrophe, code each ampersand or apostrophe as two consecutive ampersands or apostrophes. Each pair of consecutive ampersands or apostrophes counts as one character.

Subparameter Definition

delimiter
Specifies two characters that indicate the end of this data set in the input stream.

DD: DLM

Default

If you do not specify a DLM parameter, the default is the /* delimiter statement.

If the system finds an error on the DD statement before the DLM parameter, it does not recognize the value assigned as a delimiter. The system reads records until it reads a record beginning with /* or //.

Relationship to Other Parameters

Code the DLM parameter only on a DD statement with the * or DATA parameter.

The DLM parameter has meaning only on statements defining data in the input stream, that is, DD * and DD DATA statements. If DLM is specified on any other statement, a JCL error message is issued.

Invalid Delimiters

If the delimiter is not two characters:

- **For JES2**, if only one character is specified, JES2 uses the installation-defined default. If more than two characters are specified, JES2 terminates the job.
- **For JES3**, if an incorrect number of characters is coded, JES3 terminates the job.

Example of the DLM Parameter

```
//DD1   DD  *,DLM=AA
        .
        .
        data
        .
AA
```

The DLM parameter assigns the characters AA as the delimiter for the data defined in the input stream by DD statement DD1. For JES2, the characters // would also serve as valid delimiters since a DD * statement was used. JES3 accepts only the characters specified for the DLM parameter as a terminator for DD * or DD DATA.

DSID Parameter

Parameter Type: Keyword, optional

Purpose: Use the DSID parameter to specify the data set identifier of an input or output data set on a diskette of the 3540 Diskette Input/Output Unit.

An input data set is read from a 3540 diskette by a diskette reader program, and an output data set is written on a 3540 diskette by a diskette writer, which is an external writer. Neither JES2 nor JES3 can read or write diskette data sets.

To read a data set from a 3540 diskette, the DD statement must contain:

- A DSID parameter.
- An * or DATA parameter, to begin the input stream data set.

To write a data set on a 3540 diskette, the DD statement must contain:

- A DSID parameter.
- A SYSOUT parameter that specifies the output class that the diskette writer processes and the name of the diskette writer.

Also, a system command, from the operator or in the input stream, must start the diskette writer before this DD statement is processed.

Note: The system ignores the DSID parameter on a DD *, DD DATA, or a DD statement with the SYSOUT parameter, except when a diskette reader or writer processes the JCL.

References: For more information about associated data sets, see *IBM 3540 Programmer's Reference*. External writers are described in *SPL: System Modifications*.

Syntax:

```
DSID={id      }
     {(id,[V])}
```

You can omit the parentheses if you code only an id.

Subparameter Definition

id

Specifies the data set identifier. The id is 1 through 8 characters. The characters must be alphanumeric, national ($, #, @), a hyphen, or a left bracket. The first character must be alphabetic or national ($, #, @).

V

Indicates that the data set label must have been previously verified on a 3741 Data Station/Workstation. This subparameter is required only on a SYSIN DD statement.

DD: DSID

Relationship to Other Parameters

Do not code the following parameters with the DSID parameter.

BURST	FLASH
CHARS	MODIFY
DDNAME	MVSGP
DYNAM	QNAME

For 3540 Diskette Input/Output Units: A DSID parameter on a DD *, DD DATA, or sysout DD statement is ignored except when detected by a diskette reader as a request for an associated data set. See *IBM 3540 Programmer's Reference*.

On a DD * or DD DATA statement processed by a diskette reader, you can specify DSID, VOLUME=SER, BUFNO, and LRECL to indicate that a diskette data set is to be merged into the input stream following the DD statement.

Example of the DSID Parameter

```
//JOB1     JOB   ,,MSGLEVEL=(1,1)
//STEP     EXEC  PGM=AION
//SYSIN    DD    *,DSID=(ABLE,V),VOLUME=SER=123456,
//             DCB=LRECL=80
//SYSPRINT DD    SYSOUT=E,DCB=LRECL=128,DSID=BAKER
```

In this example, the SYSIN DD statement indicates that the input is on diskette 123456 in data set ABLE and must have been verified. The output will be written on a diskette in data set BAKER.

DSNAME Parameter

Parameter Type: Keyword, optional

Purpose: Use the DSNAME parameter to specify the name of a data set. For a new data set, the specified name is assigned to the data set; for an existing data set, the system uses the name to locate the data set.

References: Partitioned data sets are described in *Data Administration Guide*.

Syntax:

```
{DSNAME} =name
{DSN    }

name for permanent data set:

     dsname
     dsname(member)
     dsname(generation)
     dsname(area)

name for temporary data set:

     &&dsname
     &&dsname(member)
     &&dsname(area)

name copied from earlier DD statement:

     *.ddname
     *.stepname.ddname
     *.stepname.procstepname.ddname

name for dummy data set:

     NULLFILE
```

- You can abbreviate DSNAME as DSN.
- If the data set name begins with a blank character, the system assigns the data set a temporary data set name.
- The system ignores blank characters at the end of a data set name.
- Blanks can be included in a data set name if the name is enclosed in apostrophes, such as DSNAME='AB CD'.

Special Characters: When a data set name contains special characters that are not significant to the system, other than hyphens, enclose it in apostrophes. For example, DSNAME='DS/29'.

Code each apostrophe that is part of the data set name as two consecutive apostrophes. For example, code DAYS'END as DSNAME='DAYS''END'.

The following special characters are significant to the system. Do not enclose them in apostrophes.

- Periods to indicate a qualified data set name. However, you must enclose in apostrophes a period immediately before a right parenthesis, immediately after a left parenthesis, or immediately before a comma; for example, DSNAME='(.ABC)' and DSNAME='(ABC.)' and DSNAME='A.B.C.',
- Double ampersands to identify a temporary data set name. Note that if you use apostrophes, DSNAME='&&AB' and DSNAME='&AB' refer to the same data set.
- Parentheses to enclose the member name of a partitioned data set, the area name of an indexed sequential data set, or the generation number of a generation data set.
- Plus (+) or minus (-) sign to identify a generation of a generation data group.
- The asterisk to indicate a backward reference.

On a DD statement in a cataloged or in-stream procedure, if the data set name is a symbolic parameter, do not enclose it in apostrophes. If it is enclosed in apostrophes, the system performs correct substitution only if the symbolic parameter enclosed in apostrophes is preceded by a symbolic parameter not enclosed in apostrophes.

The data set name should not contain the 44 special characters (X'04') created by the 12-4-9 multiple punch or any operation that converts the value of characters to X'04'.

Subparameter Definition

Data Set Name for Permanent Data Set

Assign a permanent data set either an unqualified or qualified name:

Unqualified Name: 1 through 8 alphanumeric or national ($, #, @) characters, a hyphen, or a character X'C0'. The first character must be alphabetic or national ($, #, @). Do not use hyphens in data set names for RACF-protected data sets. Do not use national ($, #, @) characters in ISO/ANSI/FIPS version 3 tape data set names. For example, DSNAME=ALPHA is an unqualified data set name.

Qualified Name: multiple names joined by periods. Each name is coded like an unqualified name; therefore, the name must contain a period every 8 characters or less. For example, DSNAME=ALPHA.PGM is a qualified data set name. The maximum length of a qualified data set name is:

- 44 characters, including periods.
- For a generation data group, 35 characters, including periods.
- For an output tape data set, 17 characters, including periods. If longer than 17 characters, only the rightmost 17 characters are written to the tape header label. For more information, see *Magnetic Tape Labels and File Structure Administration.*

Name for RACF-Protected Data Set: The Resource Access Control Facility (RACF) expects the data set name to have a high-level qualifier that is defined to RACF. See the *RACF Security Administrator's Guide* for details. RACF Version 1 Release 7 uses the entire data set name, from 1 through 44 characters, when protecting a tape data set.

dsname
Specifies a data set name.

dsname(member)
Specifies a permanent partitioned data set name and the name of a member within that data set.

dsname(generation)
Specifies the name of a generation data group (GDG) and the generation number (zero or a signed integer) of a generation data set within the GDG.

To retrieve all generations of a generation data group, omit the generation number.

dsname(area)
Specifies the name of a permanent indexed sequential data set and an area of the data set. The area-name is INDEX, PRIME, or OVFLOW.

If you define an indexed sequential data set on only one DD statement, omit the area name or code it as PRIME. For example, DSNAME=dsname or DSNAME=dsname(PRIME).

To retrieve an indexed sequential data set, omit the area name.

Data Set Name for Temporary Data Set

A temporary data set is created and deleted within a job. When defining a temporary data set, you can code the DSNAME parameter or omit it; if omitted, the system will generate a name for the data set.

SMS manages a temporary data set if you specify a storage class (via the DD STORCLAS parameter) or if an installation-written automatic class selection (ACS) routine selects a storage class for the temporary data set.

When coded, the data set name for a temporary data set consists of two ampersands (&&) followed by 1 through 8 alphanumeric or national ($, #, @,) characters, a hyphen, or a character X'C0'. The first character following the ampersands must be alphabetic or national ($, #, @).

The system generates a qualified name for the temporary data set. The name begins with SYS and includes the job name, the data set name from the DSNAME parameter, if coded, and other identifying characters. If several jobs enter the system at the same time and contain DD statements with the same temporary data set name or with no data set name, the qualified names generated by the system will not be unique.

Note: A single ampersand before a data set name in a cataloged or in-stream procedure signifies a symbolic parameter. However, if no value is assigned to the name on either the EXEC statement that calls the procedure or on a PROC statement in the procedure, the system treats the name as a temporary data set name.

&&dsname
Specifies the name of a temporary data set.

&&dsname(member)
Specifies the name of a temporary partitioned data set and a member within that data set.

&&dsname(area)
Specifies the name of a temporary indexed sequential data set and an area of the data set. The area name is INDEX, PRIME, or OVFLOW.

If you define an indexed sequential data set on only one DD statement, omit the area name or code it as PRIME. For example, DSNAME=&&dsname or DSNAME=&&dsname(PRIME).

Data Set Name Copied from Earlier DD Statement

A backward reference is a reference to an earlier statement in the job or in a cataloged or in-stream procedure called by this or an earlier job step. A backward reference can be coded in the DSNAME parameter to copy a data set name from an earlier DD statement.

When copying the data set name, the system also copies the following from the DD statement:

- Whether or not the data set is a PDS.
- Whether or not the data set is a temporary data set.

***.ddname**
Asks the system to copy the data set name from earlier DD statement ddname.

***.stepname.ddname**
Asks the system to copy the data set name from DD statement, ddname, in an earlier step, stepname, in the same job.

***.stepname.procstepname.ddname**
Asks the system to copy the data set name from a DD statement in a cataloged or in-stream procedure. Stepname is the name of this job step or an earlier job step that calls the procedure, procstepname is the name of the procedure step that contains the DD statement, and ddname is the name of the DD statement.

Data Set Name for Dummy Data Set

NULLFILE
Specifies a dummy data set. NULLFILE has the same effect as coding the DD DUMMY parameter.

Relationship to Other Parameters

Do not code the following parameters with the DSNAME parameter.

*	DATA	MODIFY
BURST	DDNAME	QNAME
CHARS	DYNAM	SYSOUT
COPIES	FLASH	

Do not code the DCB IPLTXID subparameter with the DSNAME parameter.

With DD AMP Parameter: When you code an AMP parameter for a VSAM data set, do not code a DSNAME:

- That contains parentheses, a minus (hyphen), or a plus (+) sign.
- That is in the form for ISAM.
- That is in the form for PAM (partitioned access method).
- That names a generation data group.

Examples of the DSNAME Parameter

```
//DD1   DD  DSNAME=ALPHA,DISP=(,KEEP),
//          UNIT=3420,VOLUME=SER=389984
```

DD statement DD1 defines a new data set and names it ALPHA. DD statements in later job steps or jobs may retrieve this data set by specifying ALPHA in the DSNAME parameter, unit information in the UNIT parameter, and volume information in the VOLUME parameter.

```
//DDSMS1   DD  DSNAME=ALPHA.PGM,DISP=(NEW,KEEP),DATACLAS=DCLAS1,
//             MGMTCLAS=MCLAS1,STORCLAS=SCLAS1
```

DD statement DDSMS1 defines a new SMS-managed data set and names it ALPHA.PGM. DD statements in later job steps or jobs may retrieve this data set by specifying ALPHA.PGM in the DSNAME parameter.

```
//DD2   DD  DSNAME=LIB1(PROG12),DISP=(OLD,KEEP),UNIT=3350
//          VOLUME=SER=882234
```

DD statement DD2 retrieves member PROG12 from the partitioned data set named LIB1.

```
//DD3   DD  DSNAME=&&WORK,UNIT=3420
```

DD statement DD3 defines a temporary data set. Because the data set is deleted at the end of the job step, the DSNAME parameter can be omitted. The following example shows why a temporary data set should be named.

```
//STEP1  EXEC PGM=CREATE
//DD4    DD   DSNAME=&&ISDATA(PRIME),DISP=(,PASS),UNIT=(3350,2),
//       VOLUME=SER=334859,SPACE=(CYL,(10,,2),,CONTIG),DCB=DSORG=IS
//STEP2  EXEC PGM=OPER
//DD5    DD   DSNAME=*.STEP1.DD4,DISP=(OLD,DELETE)
```

DD statement DD4 in STEP1 defines a temporary indexed sequential data set named ISDATA. This DD statement defines all of the areas of an indexed sequential data set. DD statement DD5 in STEP2 retrieves the data set by referring to the earlier DD statement that defines the data set. Because the temporary data set is passed when it is defined in STEP1, it is not deleted at the end of STEP1 and STEP2 can retrieve it.

DUMMY Parameter

Parameter Type: Positional, optional

Purpose: Use the DUMMY parameter to specify that:

- No device or external storage space is to be allocated to the data set.
- No disposition processing is to be performed on the data set.
- For BSAM and QSAM, no input or output operations are to be performed on the data set.

One use of the DUMMY parameter is in testing a program. When testing is finished and you want input or output operations performed on the data set, replace the DD DUMMY statement with a DD statement that fully defines the data set.

Another use of the DUMMY parameter is in a cataloged or in-stream procedure. Code on the DD DUMMY statement all the required parameters. When the procedure is called, nullify the effects of the DUMMY parameter by coding on the DD statement that overrides the DD DUMMY statement a DSNAME parameter that matches the DSNAME parameter on the DD DUMMY statement. For example, procedure step PS contains the following:

```
//DS1      DD     DUMMY,DSNAME=A,DISP=OLD
```

Nullify the DUMMY parameter by coding:

```
//JS       EXEC   PROC=PROC1
//PS.DS1   DD     DSNAME=A
```

Syntax:

```
//ddname  DD  DUMMY[,parameter]...
```

All parameters coded on a DD DUMMY statement must be syntactically correct. The system checks their syntax.

Parameters on DD DUMMY Statements

- Code the DUMMY parameter by itself or follow it with all the parameters you would normally code when defining a data set, except the DDNAME parameter.
- Code the DCB parameter, if needed. If the program does not supply all the data control block information, make sure that the DCB parameter supplies the missing information.
- Code AMP=AMORG if you are using VSAM.
- If you code either VOLUME=REF=dsname or DCB=dsname with DUMMY, the referenced dsname must be cataloged or passed; otherwise, the job is terminated.
- Because no I/O is performed to the dummy data set, the system checks the UNIT, SPACE, and DISP parameters, if coded, for syntax, then ignores them.

Relationship to Other Parameters

Do not code the following parameters with the DUMMY parameter.

*	DYNAM
DATA	QNAME
DDNAME	

Relationship to Other Control Statements

Backward References: If a later DD statement in a job refers to a DD DUMMY statement when requesting unit affinity (UNIT=AFF=ddname) or volume affinity (VOLUME=REF=*.stepname.ddname), the system assigns a dummy status to the later DD statement.

Overriding a Procedure DD Statement: Coding DUMMY on a DD statement that overrides a DD statement in a procedure does not nullify symbolic parameters on the overridden DD statement. You must assign values to, or nullify, symbolic parameters on the overridden DD statement as described in "Symbolic Parameters" on page 5-8.

The DSNAME parameter on the overriding DD statement must not specify NULLFILE.

If the overriding DD statement contains a SUBSYS parameter, the system nullifies a DUMMY parameter on the overridden DD statement in the procedure.

Data Sets Concatenated to Dummy Data Sets: The system treats data sets concatenated to a DUMMY data set as dummy data sets in that I/O operations are bypassed. However, the system performs disposition processing and allocates devices and storage for any concatenated data sets.

Relationship to Access Methods

Use one of the following access methods with the DUMMY parameter:

- Basic sequential access method (BSAM)
- Virtual storage access method (VSAM)
- Queued sequential access method (QSAM)
- BDAM load mode (BSAM with MACRF=WL in the data control block)

If you use any other access method, the job is terminated.

Examples of the DUMMY Parameter

```
//OUTDD1  DD  DUMMY,DSNAME=X.X.Z,UNIT=3350,
//            SPACE=(TRK,(10,2)),DISP=(,CATLG)
```

DD statement OUTDD1 defines a dummy data set. The other parameters coded on the statement are checked for syntax but not used.

```
//IN1  DD  DUMMY,DCB=(BLKSIZE=800,LRECL=400,RECFM=FB)
```

DD statement IN1 defines a dummy data set. The DCB parameter supplies data control block information not supplied in the program. Without it, the step might be abnormally terminated.

```
//IN2 DD  DUMMY,DSNAME=ELLN,DISP=OLD,VOLUME=SER=11257,UNIT=3350
```

When calling a cataloged procedure that contains DD statement IN2 in procedure step STEP4, you can nullify the effects of the DUMMY parameter by coding:

```
//STEP4.IN2  DD  DSNAME=ELLN
```

```
//TAB  DD  DSNAME=APP.LEV12,DISP=OLD
```

If you call a cataloged procedure that contains DD statement TAB in procedure step STEP1, you can make this DD statement define a dummy data set by coding:

```
//STEP1.TAB  DD  DUMMY
```

```
//MSGS  DD  SYSOUT=A
```

If you call a cataloged procedure that contains the DD statement MSGS in procedure step LOCK, you can make this DD statement define a dummy data set by coding:

```
//LOCK.MSGS  DD  DUMMY
```

DYNAM Parameter

Parameter Type: Positional, optional

Purpose: Use the DYNAM parameter to increase by one the control value for dynamically allocated resources held for reuse. Even when DYNAM is not coded, the system normally holds resources in anticipation of reuse. The DYNAM parameter is supported to provide compatibility with older systems.

A DD DYNAM statement is a dummy request.

Syntax:

```
//ddname  DD  DYNAM   [comments]
```

Relationship to Other Parameters

Do not code any parameters with the DYNAM parameter.

Do not code on a DD DYNAM statement a ddname that is meaningful to the system; for example, JOBLIB, SYSCHK.

Relationship to Other Control Statements

- Do not refer to a DD DYNAM statement in a DDNAME parameter.
- To nullify the DYNAM parameter on a DD statement in a cataloged or in-stream procedure, code a SYSOUT or DSNAME parameter in the overriding DD statement. DSNAME=NULLFILE does not nullify a DYNAM parameter.
- Do not code a backward reference to a DD DYNAM statement.
- Do not code the DYNAM parameter on the first DD statement for a concatenation.

Example of the DYNAM Parameter

```
//INPUT  DD  DYNAM
```

This DD statement increases by one the control value for dynamically allocated resources held for reuse.

EXPDT Parameter

Parameter Type: Keyword, optional

Purpose: Use the EXPDT parameter to specify the expiration date for a new data set. On and after the expiration date, the data set can be deleted or written over by another data set.

If the DD statement contains DISP=(NEW,DELETE) or the DISP parameter is omitted to default to NEW and DELETE, the system deletes the data set when the step terminates normally or abnormally, even though an expiration date is also specified.

Do not specify EXPDT for a temporary data set.

The EXPDT parameter achieves the same result as the RETPD parameter.

Code the EXPDT parameter when you want to (1) specify an expiration date for the data set or (2) with SMS, override the expiration date defined in the data class for the data set.

Note: Other than information for SMS, this information for the EXPDT parameter previously appeared under the LABEL EXPDT subparameter.

Syntax:

```
EXPDT= {yyddd    }
       {yyyy/ddd }
```

Subparameter Definition

EXPDT=yyddd
EXPDT=yyyy/ddd
Specifies an expiration date for the data set.

yyddd
The yy is a two-digit year number (through 99) and the ddd is a three-digit day number from 001 through 366. For example, code February 2, 1996 as EXPDT=96033. (For expiration dates of January 1, 2000 and later, you must use the form EXPDT=yyyy/ddd.)

Note: Expiration dates of 99365 and 99366 are considered "never-scratch" dates. Data sets with these expiration dates are not deleted or written over.

yyyy/ddd
The yyyy is a four-digit year number (through 2155) and the ddd is a three-digit day number from 001 through 366. For example, code February 2, 1996 as EXPDT=1996/033, and code February 2, 2006 as EXPDT=2006/033.

Note: Expiration dates of 1999/365 and 1999/366 are considered "never-scratch" dates. Data sets with these expiration dates are not deleted or written over.

The years from 1900 can be specified. However, if you specify the current date or an earlier date, the data set is immediately eligible for replacement.

Overrides

With SMS, EXPDT overrides the expiration date defined in the data class for the data set.

With SMS, both the expiration date specified on EXPDT and defined in the data class for an SMS-managed data set can be limited by a maximum expiration date defined in the management class for the data set.

Relationship to Other Parameters

Do not code the following DD parameters with the EXPDT parameter.

*	DYNAM
DATA	RETPD
DDNAME	SYSOUT

Deleting a Data Set Before its Expiration Date

To delete a data set before the expiration date has passed, use one of the following:

- For data sets cataloged in a VSAM or integrated catalog facility catalog, use the DELETE command, as described in *Integrated Catalog Administration: Access Method Services Reference* or *VSAM Catalog Administration: Access Method Services Reference*.
- For data sets not cataloged in a VSAM or integrated catalog facility catalog, use the IEHPROGM utility, as described in *Data Administration: Utilities*.
- For the data set control block (DSCB), use the SCRATCH macro with the OVRD parameter, as described in *System-Data Administration*. Deletion of the DSCB makes the space occupied by the data set available for reallocation.

Examples of the EXPDT Parameter

```
//DD7  DD  DSNAME=TOM1,DISP=(NEW,KEEP),EXPDT=2006/033,
//         UNIT=3350,SPACE=(TRK,(1,1)),VOLUME=SER=663344
```

In the example, the data set is not eligible for being deleted or written over until February 2, 2006.

```
//SMSDS2  DD  DSNAME=MYDS2.PGM,DATACLAS=DCLAS02,DISP=(NEW,KEEP),
//            EXPDT=1996/033
```

In the example, the expiration date of February 2, 1996, overrides the expiration date defined in the data class for the data set.

FCB Parameter

Parameter Type: Keyword, optional

Purpose: Use the FCB parameter to specify:

- The forms control buffer (FCB) image JES is to use to guide printing of this sysout data set by a 1403 Printer, 3211 Printer, 3203 Printer Model 5, 3800 Printing Subsystem, 4245 Printer, 4248 Printer, or by a printer supported by systems network architecture (SNA) remote job entry (RJE).

- The carriage control tape JES is to use to control printing of this sysout data set by a 1403 Printer or by a printer supported by SNA RJE.

- The data-protection image JES is to use to control output of this sysout data set by a 3525 Card Punch.

The FCB image specifies how many lines are to be printed per inch and the length of the form. JES loads the image into the printer's forms control buffer. The FCB image is stored in SYS1.IMAGELIB. IBM provides three standard FCB images:

- STD1, which specifies 6 lines per inch on an 8.5-inch-long form. (3211 and 3203-2 only)
- STD2, which specifies 6 lines per inch on an 11-inch-long form. (3211 and 3203-2 only)
- STD3, which in a JES3 system specifies 8 lines per inch for a dump. (3800 only)

References: For more information on the forms control buffer, see:

MVS/XA System-Data Administration,
Programming Support for the IBM 3505 Card Reader and IBM 3525 Card Punch, or
3800 Printing Subsystem Programmer's Guide.

Syntax:

```
FCB={fcb-name                          }
    {(fcb-name[,ALIGN|,VERIFY])        }
```

- You can omit the parentheses if you code only the fcb-name.
- Code the fcb-name as STD1 or STD2 only to request the IBM-supplied images.
- Code the fcb-name as STD3 only for a high-density dump in a JES3 system.

Subparameter Definition

fcb-name
Identifies the FCB image. The name is 1 through 4 alphanumeric or national ($, #, @) characters and is the last characters of a SYS1.IMAGELIB member name:

- FCB2xxxx member for a 3211, a 3203 model 5, or a printer supported by SNA.
- FCB3xxxx member for a 3800.
- FCB4xxxx member for a 4248.

ALIGN
Requests that the system ask the operator to check the alignment of the printer forms before the data set is printed.

Note:

- ALIGN is ignored for a sysout data set.
- ALIGN is ignored for a data set printed on a 3800. The 3800 does not use the ALIGN subparameter.

VERIFY
Requests that the system ask the operator to verify that the image displayed on the printer is for the desired FCB image. The operator can also take this opportunity to align the printer forms.

Note: VERIFY is ignored for a sysout data set.

Defaults

If you do not code the FCB parameter, the system checks the FCB image in the printer's forms control buffer; if it is a default image, as indicated by its first byte, JES uses it. If it is not a default image, JES loads the FCB image that is the installation default specified at JES initialization.

Overrides

An FCB parameter on a sysout DD statement overrides an OUTPUT JCL FCB parameter.

Relationship to Other Parameters

Do not code the following parameters with the FCB parameter.

*	DYNAM
AMP	KEYOFF
DATA	PROTECT
DDNAME	QNAME

Do not code the following DCB subparameters with the FCB parameter.

CYLOFL	INTVL
RKP	

For output to the 3525, do not code the SYSOUT parameter and the FCB parameter; the system ignores the FCB parameter.

Relationship to Other Control Statements

You can also code the FCB parameter on the following:

- The OUTPUT JCL statement.
- The JES2 /*OUTPUT statement.
- The JES3 //*FORMAT PR statement.

Defining an FCB Image for a Work Station

When a work station uses a peripheral data set information record (PDIR), the FCB image is defined in the work station. The DD statement FCB fcb-name subparameter must match the FCB name defined in the PDIR work station.

When a work station does not use a PDIR, add an FCB member to SYS1.IMAGELIB. At setup time, JES3 translates the FCB into a set vertical format (SVF).

Requesting a High-Density Dump

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 of the FCB Parameter

```
//DD1    DD  UNIT=3211,FCB=(IMG1,VERIFY)
```

In this example, the DD statement defines an output data set to be printed by a 3211. The FCB parameter requests that the data set be printed under control of the FCB2IMG1 member in SYS1.IMAGELIB. Because VERIFY is coded, the system displays the FCB image on the printer before printing the data set.

```
//DD2    DD  SYSOUT=A,FCB=IMG2
```

This sysout DD statement specifies output class A. If output class A routes output to a printer having the forms control buffer feature, JES loads the FCB image IMG2 into the forms control buffer. If the printer does not have the forms control buffer feature, the operator receives a message to mount the carriage control tape IMG2 on the printer.

```
//OUTDDS   DD   UNIT=3211,FCB=(6,ALIGN)
```

In this example, the DD statement defines an output data set to be printed by a 3211. The FCB parameter requests that the data set be printed under control of the FCB image named 6. Because ALIGN is coded, the system issues a message to the operator requesting that the alignment of the printer forms be checked before the data set is printed.

```
//PUNCH    DD   UNIT=3525,FCB=DP2
```

In this example, the DD statement requests output on a 3525. Therefore, the FCB parameter defines the data protection image to be used for the 3525.

```
//SYSUDUMP   DD   SYSOUT=A,FCB=STD3
```

In this example, the DD statement requests that the 3800 print a dump at 8 lines per inch.

FLASH Parameter

Parameter Type: Keyword, optional

Purpose: Use the FLASH parameter to identify the forms overlay to be used in printing this sysout data set on a 3800 Printing Subsystem and, optionally, to specify the number of copies on which the forms overlay is to be printed.

Note: FLASH applies only for a data set printed on a 3800.

References: For information on forms overlays, see the *Forms Design Reference Guide for the IBM 3800 Printing Subsystem.*

Syntax:

```
        ┌overlay-name          ┐
FLASH=  │(overlay-name[,count]) │
        └NONE                  ┘
```

The count subparameter is optional. If you omit it, you can omit the parentheses. However, if you omit it, you must not code it as a null; for example, FLASH=(ABCD,) is invalid.

Subparameter Definition

overlay-name
Identifies the forms overlay frame that the operator is to insert into the printer before printing begins. The name is 1 through 4 alphanumeric or national ($, #, @) characters.

count
Specifies the number, 0 through 255, of copies that JES is to flash with the overlay, beginning with the first copy printed. Code a count of 0 to flash **all** copies.

NONE
Suppresses flashing for this sysout data set.

If FLASH=NONE is on a DD statement in a job to be executed at a remote node, JES3 sets the overlay-name to zero before sending the job to the node.

Defaults

If you do not code a FLASH parameter or an installation default was not specified at JES2 or JES3 initialization, forms are not flashed.

If you specify an overlay-name without specifying a count or with a count of 0, all copies are flashed. That is, the default for count is 255.

Overrides

A FLASH parameter on a sysout DD statement overrides an OUTPUT JCL FLASH parameter.

Note: A null first subparameter is invalid in a FLASH parameter on a DD statement, but is permitted on an OUTPUT JCL statement.

Relationship to Other Parameters

Do not code the following parameters with the FLASH parameter.

*	DSID	PROTECT
AMP	DSNAME	QNAME
DATA	DYNAM	SUBSYS
DDNAME	LABEL	VOLUME
DISP	MSVGP	

Relationship to COPIES Parameter: If this DD statement or a referenced OUTPUT JCL statement also contains a COPIES parameter, JES prints with the forms overlay the number of copies specified in one of the following:

- COPIES=nnn, if the FLASH count is larger than nnn. For example, if COPIES=10 and FLASH=(LTHD,12) JES prints 10 copies, all with the forms overlay.
- The sum of the group-values specified in the COPIES parameter, if the FLASH count is larger than the sum. For example, if COPIES=(,(2,3,4)) and FLASH=(LTHD,12) JES prints nine copies in groups, all with the forms overlay.
- The count subparameter in the FLASH parameter, if the FLASH count is smaller than nnn or the sum from the COPIES parameter. For example, if COPIES=10 and FLASH=(LTHD,7) JES prints seven copies with the forms overlay and three copies without.

Relationship to Other Control Statements

FLASH can also be coded on the following:

- The OUTPUT JCL statement.
- The JES3 //*FORMAT PR statement.
- The JES2 /*OUTPUT statement.

Verification of Forms Overlay Frame

Before printing starts, the system requests the operator to load the specified forms overlay frame in the printer. A frame must be loaded, but the system cannot verify that it is the correct frame.

DD: FLASH

Printing without Flashing

To print without flashing, specify one of the following:

- FLASH=NONE on the DD or OUTPUT JCL statement.
- Omit the FLASH parameter on all of the statements for the data set and on all JES initialization statements.
- For a sysout data set, omit the FLASH parameter on the DD statement and specify FLASH=(,0) on a referenced OUTPUT JCL statement.

Example of the FLASH Parameter

```
//DD1 DD  SYSOUT=A,COPIES=10,FLASH=(ABCD,5)
```

In this example, JES issues a message to the operator requesting that the forms-overlay frame named ABCD be inserted into the printer. Then JES prints the first five copies of the data set with the forms-overlay and the last five copies without.

FREE Parameter

Parameter Type: Keyword, optional

Purpose: Use the FREE parameter to specify when the system is to deallocate the resources used for this DD statement's data set. The resources can be devices, volumes, or exclusive use of a data set.

Use FREE=CLOSE on a sysout DD statement to make JES print the sysout data set before the job is finished.

Syntax:

```
FREE=(END  )
      (CLOSE)
```

Subparameter Definition

END
Requests that the system deallocate the data set at the end of the step.

CLOSE
Requests that the system deallocate the data set when it is closed.

Defaults

If no FREE parameter is specified, the default is END. Also, if the FREE parameter is incorrectly coded, the system substitutes END and issues a warning message.

Overrides

FREE=CLOSE is ignored when:

- The data set is a member of a concatenated group.
- The task using the data set abnormally terminates.
- The data set is referenced by another DD statement in the same or subsequent step.

Relationship to Other Parameters

Do not code the following parameters with the FREE parameter.

*	DYNAM
DATA	QNAME
DDNAME	

If the DD statement specifies FREE=END and a DISP subparameter of PASS, the data set is not deallocated until the end of the job or until used for a later DD statement with a disposition of other than PASS.

Do not specify FREE=CLOSE on a DD statement with a ddname of JOBCAT, JOBLIB, STEPCAT, or STEPLIB; CLOSE is ignored.

When you specify FREE=CLOSE and the job step abnormally terminates before the data set is closed, the system uses the abnormal termination disposition from the DISP parameter to process the data set. However, when you specify FREE=CLOSE and the job step abnormally terminates after the data set is closed, then the system has already processed the data set using the normal termination disposition.

Relationship to Other Control Statements

If a DD statement requests unit affinity in a UNIT=AFF parameter or volume affinity in a VOLUME=REF parameter with an earlier DD statement, do not code FREE=CLOSE on the earlier statement.

If you code FREE=CLOSE on a sysout DD statement that references an OUTPUT JCL statement containing a GROUPID parameter, JES2 will not group the data sets into one output group. Instead, JES2 produces one copy of the sysout data set for each OUTPUT JCL statement that the DD statement references.

Relationship to the CLOSE Macro Instruction

When FREE=CLOSE is specified for a data set that is opened and closed more than once during a job step:

- The data set is deallocated after it is closed if the assembler CLOSE macro instruction specifies DISP, REWIND, or FREE. If the data set is reopened after the system has deallocated it, the job step abnormally terminates, unless the data set is dynamically allocated in the interval.
- The data set is not deallocated until the end of the job step if the assembler CLOSE macro instruction specifies LEAVE or REREAD. Then the data set can be reopened.

Examples of the FREE Parameter

```
//EA33  DD  SYSOUT=D,FREE=CLOSE
```

In this example, the FREE=CLOSE parameter makes JES deallocate this output class D data set when it is closed, rather than at the end of the job step. JES schedules the data set for printing.

```
//EA33  DD  DSNAME=SYBIL,DISP=OLD,FREE=CLOSE
```

In this example, the FREE=CLOSE parameter makes JES deallocate the data set, dequeue it, and make it available to other jobs as soon as it is closed.

```
//STEP1    EXEC PGM=ABLE1
//DD1      DD   DSNAME=A,DISP=(,PASS),FREE=END
//STEP2    EXEC PGM=ABLE2
//DD2      DD   DSNAME=A,DISP=(OLD,CATLG),FREE=END
```

In this example, data set A is passed by STEP1 to STEP2. FREE=END on DD statement DD1 is ignored because the disposition is PASS. FREE=END on DD statement DD2 causes data set A to be deallocated at the end of STEP2, when it is also cataloged.

```
//STEP1    EXEC PGM=BAKER1
//DD       DD   DSNAME=A,DISP=(NEW,PASS),FREE=END
//STEP2    EXEC PGM=BAKER2
```

In this example, data set A is a new data set. Because PASS is specified, FREE=END is ignored and the data set remains allocated.

HOLD Parameter

Parameter Type: Keyword, optional

Purpose: Use the HOLD parameter to tell the system to hold a sysout data set until it is released by the system operator. When the data set is ready for processing, notify the system operator to release it via a TSO NOTIFY parameter, a JES2 /*MESSAGE statement, or a JES3 //*OPERATOR statement.

A TSO user can specify HOLD=YES in order to retrieve a sysout data set and display it on a terminal. For JES3, only work on the hold queue can be processed by the TSO user.

Note: HOLD is supported only for sysout data sets. If HOLD appears on a DD statement that does not contain a SYSOUT parameter, it is ignored.

Syntax:

```
HOLD= (YES )
      (Y   )
      (NO  )
      (N   )
```

Subparameter Definition

YES
Requests that the system hold the sysout data set until the data set is released by the system operator. This subparameter can also be coded as Y.

NO
Requests that the system perform installation-defined processing for the sysout data set's output class. This subparameter can also be coded as N.

Defaults

If no HOLD parameter is specified, the default is NO. If the HOLD parameter is incorrectly coded, the system assumes the default of NO and issues a warning message; the job continues.

Overrides

HOLD=NO is overridden by the deallocation verb of dynamic allocation or the TSO FREE command.

Relationship to Other Parameters

Code the HOLD parameter only on a DD statement with the SYSOUT parameter.

JES3 ignores HOLD=YES when

- DEST=(node,userid) is coded, or
- the sysout data set is placed on the hold queue, for example, if SYSOUT=(,writer-name) is coded.

Relationship to Other Control Statements

Code a NOTIFY parameter on the JOB statement to ask the system to send a message to your TSO userid when job processing is complete.

JES2 users can use the /*NOTIFY control statement to direct job notification messages and to override a JOB NOTIFY parameter.

Example of the HOLD Parameter

```
//JOB01  JOB  ,'HAROLD DUQUETTE',MSGLEVEL=1
//STEP1  EXEC PGM=MJCOSCO
//DD1    DD   SYSOUT=B,DEST=RMT6,HOLD=YES
```

Sysout data set DD1 from JOB01 is held on a queue until the TSO user at RMT6 asks the system operator to release the data set.

KEYLEN Parameter

Parameter Type: Keyword, optional

Purpose: Use the KEYLEN parameter to specify the length of the keys used in a data set.

Code the KEYLEN parameter when you want to (1) specify a key length for the data set or (2) with SMS, override the key length defined in the data class of the data set.

For an existing data set, the key length can be supplied from the data set label (or data class with SMS). If a key length is not specified or supplied, input or output requests must not require keys.

KEYLEN applies to data sets with the BDAM, BPAM, BSAM, EXCP, QISAM, and TCAM access methods, and, with SMS, to VSAM data sets.

Note: Other than information for SMS, this information for the KEYLEN parameter previously appeared under the DCB KEYLEN subparameter.

Syntax:

```
KEYLEN=bytes
```

Subparameter Definition

bytes
Specifies the length, in bytes, of the keys used in the data set.

The number of bytes is:

- 0 to 255 for non-VSAM data sets. The key length must be less than or equal to the record length.
- 1 to 255 for VSAM key-sequenced (RECORG=KS) data sets. A key length must be specified, either explicitly with the KEYLEN or LIKE parameter, or in the data class for the data set. The key length must be less than the record length.

Overrides

KEYLEN overrides the key length specified in the data set label, and with SMS, KEYLEN overrides the key length defined in the data class for the data set.

Relationship to Other Parameters

Do not code the following DD parameters with the KEYLEN parameter.

*	DCB=STACK
DATA	DCB=TRTCH
DCB=KEYLEN	DDNAME
DCB=MODE	DYNAM
DCB=PRTSP	

Examples of the KEYLEN Parameter

```
//DD4    DD     DSNAME=JST,DISP=(NEW,KEEP),UNIT=3350,
//              SPACE=(CYL,(12,2)),DCB=(A.B.C),KEYLEN=8
```

DD statement DD4 defines a new data set named JST and requests that the system copy the DCB information from the data set label of the cataloged data set named A.B.C. If the data set label contains a key length specification, it is overridden by the KEYLEN coded on this DD statement.

```
//SMSDS3   DD   DSNAME=MYDS3.PGM,DATACLAS=VSAM1,DISP=(NEW,KEEP),
//            KEYLEN=6
```

In the example, where the data class VSAM1 defines a key-sequenced VSAM data set, the key length of 6 overrides the key length defined in the data class.

KEYOFF Parameter

Parameter Type: Keyword, optional — use this parameter only with SMS

Without SMS, use the RKP subparameter of the DCB parameter described on page 10-57.

Purpose: Use the KEYOFF parameter to specify the key offset, the position of the first byte of the record key in each logical record of a new VSAM data set. The first byte of a logical record is position 0.

If SMS is not installed or is not active, the system syntax checks and then ignores the KEYOFF parameter.

Code the KEYOFF parameter only for a VSAM key-sequenced data set (RECORG=KS).

Code the KEYOFF parameter when you want to (1) specify a key offset for the data set or (2) override the key offset defined in the data class of the data set.

References: See *MVS/Extended Architecture VSAM Administration Guide* for information on VSAM key-sequenced data sets.

Syntax:

```
KEYOFF=offset-to-key
```

Subparameter Definition

offset-to-key
Specifies the position (offset), in bytes, of the first byte of the key in each record. The offset is 0 to the difference between the record length (LRECL) and key length (KEYLEN), in the range 0 to 32760.

Overrides

KEYOFF overrides the key offset defined in the data class for the data set.

Relationship to Other Parameters

Do not code the following DD parameters with the KEYOFF parameter.

*	DYNAM
DATA	FCB
DCB=RESERVE	UCS
DCB=RKP	
DDNAME	

Example of the KEYOFF Parameter

```
//SMSDS3   DD   DSNAME=MYDS3.PGM,DATACLAS=VSAM1,DISP=(NEW,KEEP),
//              KEYOFF=2
```

In the example, the data class VSAM1 defines a key-sequenced VSAM data set. The key offset of 2 overrides the key offset defined in the data class and specifies that the first byte of the key is in the third position of each record.

LABEL Parameter

Parameter Type: Keyword, optional

Purpose: Use the LABEL parameter to specify for a tape or direct access data set:

- The type and contents of the label or labels for the data set.
- If a password is required to access the data set.
- If the system is to open the data set only for input or output.
- The expiration date or retention period for the data set.

Although subparameters RETPD and EXPDT are shown in the syntax of the LABEL parameter, you should use the RETPD or EXPDT DD parameter to specify a retention period or expiration date for the data set.

For a tape data set, this parameter can also specify the relative position of the data set on the volume.

References: For details on tape labels, see *Magnetic Tape Labels and File Structure Administration*. For details on direct access labels, see *Data Administration Guide*. For information on protecting a data set with a password, see *System-Data Administration*.

Syntax:

```
LABEL=([data-set-sequence-number] [,label] [,PASSWORD] [,IN ] [,RETPD=nnnn      ])
       (                          [,      ] [,NOPWREAD] [,OUT] [,EXPDT={yyddd  }])
                                  [,      ]                            {yyyy/ddd}

        label is one of the following:
            SL
            SUL
            AL
            AUL
            NSL
            NL
            BLP
            LTM
```

The first four subparameters are positional; the last subparameter is keyword. If you omit any positional subparameters but code a following **positional** subparameter, indicate each omitted subparameter by a comma. If the following subparameter is keyword (EXPDT or RETPD), commas are not needed to indicate omitted subparameters. For example:

LABEL=(0001,SUL,PASSWORD,IN)
LABEL=(,SUL,PASSWORD)
LABEL=(,SUL,,IN,EXPDT=97033)
LABEL=(,,PASSWORD,EXPDT=1997/033)
LABEL=(,SUL,EXPDT=1997/033)
LABEL=(0001,,,IN)
LABEL=(0001,EXPDT=1997/033)

If you specify only the data-set-sequence-number or only the retention period or only the expiration date, you can omit the parentheses. For example, code LABEL=data-set-sequence-number, LABEL=RETPD=nnnn, LABEL=EXPDT=yyddd, or LABEL=EXPDT=yyyy/ddd.

Alternate Syntax for RETPD and EXPDT

RETPD and EXPDT should be specified as DD parameters rather than subparameters of the LABEL parameter. This allows you to specify a retention period or expiration date without the need to code LABEL.

For example, code RETPD and EXPDT on the DD statement as:

RETPD=366 or EXPDT=2006/033

See the DD RETPD parameter described on page 10-145 and the DD EXPDT parameter described on page 10-90.

Subparameter Definition

Data-Set-Sequence-Number

data-set-sequence-number
Identifies the relative position of a data set on a tape volume. The data-set-sequence-number is 1 through 4 decimal digits. Omit this subparameter or code 0 or 1 to indicate the first data set on the tape volume.

Omit this subparameter for the following:

- Cataloged data sets. The system obtains the data-set-sequence-number from the catalog.
- A DD DSNAME parameter that requests all members of a generation data group (GDG). The system retrieves the data-set-sequence-number from the catalog.
- A data set passed from a preceding step. The system obtains the data-set-sequence-number from the passing step.

Label

The system does not retain label type information for cataloged data sets; if the label type is not coded in the LABEL parameter for a cataloged data set, the system assumes SL.

For a data set on a direct access device, the system obtains the label type from the DD statement; the label type is not obtained from any other source referred to in the DD statement. Only two label types are valid for direct access devices: SL and SUL.

SL
Indicates that a data set has IBM standard labels. If this subparameter is omitted, SL is the default.

Code only SL or SUL for data sets on direct access devices.

If the LABEL parameter is coded on a SYSCKEOV DD statement, code LABEL=(,SL).

SUL
Indicates that a data set has both IBM standard and user labels.

Code only SL or SUL for data sets on direct access devices.

Do not code SUL for partitioned or indexed sequential data sets.

AL
Indicates that a tape data set has ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels.

If you specify AL for a tape generation data set for output, the ending .GnnnnVnn (where n=0 through 9) **will not** appear as part of the file identifier (data set name field) of the HDR1 label. Instead, the data is placed in the generation and version number fields of the HDR1 label.

AUL
Indicates that a tape data set has user labels and ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels.

NSL
Indicates that a tape data set has nonstandard labels.

NL
Indicates that a tape data set has no labels.

When retrieving two or more data sets from several NL or BLP tape volumes, concatenate the DD statements and repeat the LABEL parameter on each DD statement.

If you are processing ASCII data on unlabeled tapes, the data control block must specify OPTCD=Q.

BLP
Requests that the system bypass label processing for a tape data set.

If the installation did not specify the BLP feature in the reader cataloged procedure, BLP has the same effect as NL.

If you code BLP and the tape volume has labels, a tapemark delimits the data set. To let the system position a tape with labels to the proper data set, code the data-set-sequence-number subparameter; the number must reflect all labels and data sets that precede the desired data set.

Do not specify BLP when the DD DSNAME parameter requests all members of a generation data group (GDG); the system obtains the data-set-sequence-number from the catalog. Therefore, coding BLP might result in incorrect tape positioning.

When retrieving two or more data sets from several NL or BLP tape volumes, concatenate the DD statements and repeat the LABEL parameter on each DD statement.

LTM
Indicates that the data set has a leading tapemark.

Password Protection

For an SMS-managed data set (one with an assigned storage class), SMS sets the password indicators in the VTOC and catalog but ignores the indicators and does not use password protection for the data set. See the DD SECMODEL parameter described on page 10-147.

Password protecting data sets requires the following:

- Data set names no longer than 17 characters. MVS retains in the tape label only the rightmost 17 characters of the data set name. Consequently, longer names could be identical in password checks.
- Volumes with IBM standard labels or ISO/ANSI/FIPS Version 3 labels.
- A password assigned in the PASSWORD data set. If a password is not assigned, the system will abnormally terminate a job step when it attempts to open the data set for output, if NOPWREAD is coded, or for input or output, if PASSWORD is coded.

To create a password-protected data set following an existing password-protected data set, code the password of the existing data set. The password must be the same in both the existing and the new data set.

To password-protect a data set on a tape volume containing other data sets, you must password-protect all the data sets on the volume and the passwords must be the same for all data sets.

To password-protect an existing data set using PASSWORD or NOPWREAD, open the data set for output the first time it is used during the job step.

PASSWORD
Indicates that a data set cannot be read, changed, deleted, or written to unless the system operator or TSO user supplies the correct password.

NOPWREAD
Indicates that a data set cannot be changed, deleted, or written to unless the system operator or TSO user supplies the correct password. No password is necessary for reading the data set.

Input or Output Processing

IN
Indicates that a BSAM data set opened for INOUT or a BDAM data set opened for UPDAT is to be read only. The IN subparameter overrides the processing option in the assembler OPEN macro instruction. Any attempt by the processing program to write in the data set makes the system give control to the error analysis (SYNAD) routine.

OUT
Indicates that a BSAM data set opened for OUTIN or OUTINX is to be written in only. The OUT subparameter overrides the processing option in the assembler OPEN macro instruction. Any attempt by the processing program to read the data set makes the system give control to the error analysis (SYNAD) routine.

DD: LABEL

Retention Period or Expiration Date for Data Set

You should avoid using the RETPD and EXPDT subparameters on the LABEL parameter to specify a retention period or expiration date for the data set.

Use the DD RETPD parameter (described on page 10-145) or the DD EXPDT parameter (described on page 10-90), which do the same function. This allows you to specify a retention period or expiration date without the need to code the LABEL parameter.

Defaults

- If no data-set-sequence-number subparameter is specified or if the number is coded as 0 or 1, the default is the first data set on the tape volume, unless the data set is passed or cataloged.
- If no label type subparameter is specified, the default is only IBM standard labels (SL).

Relationship to Other Parameters

Do not code the following parameters with the LABEL parameter.

*	DATA	MODIFY
BURST	DDNAME	QNAME
CHARS	DYNAM	SYSOUT
COPIES	FLASH	

Do not specify the LABEL parameter with the FUNC subparameter of the DCB parameter. The results are unpredictable.

ISO/ANSI/FIPS Version 3 tape data sets can be protected by use of the ACCODE parameter.

If you specify a LABEL parameter on a SYSCKEOV DD statement, code LABEL=(,SL).

Relationship to Other Control Statements

When a VOLUME=REF subparameter refers to an earlier DD statement to use the same volume(s):

- For tape, the system copies the **LABEL** label type subparameter from the referenced DD statement; the copied label type overrides the label type on the referencing DD statement.
- For direct access, the system uses a LABEL=(,SL) or LABEL=(,SUL) subparameter from the referencing DD statement. If the referencing DD statement specifies any other label type, the system copies the LABEL label type subparameter from the referenced DD statement; the copied label type overrides the label type on the referencing DD statement.

Translation

If the installation specified ASCII=INCLUDE during system generation, then AL or AUL in the LABEL parameter requests translation. You can also request translation by specifying OPTCD=Q in the data control block. If the tape is not labeled, LABEL=(,NL), you *must* specify OPTCD=Q for translation to occur.

Examples of the LABEL Parameter

```
//DD1   DD   DSNAME=HERBI,DISP=(NEW,KEEP),UNIT=TAPE,
//           VOLUME=SER=T2,LABEL=(3,NSL,RETPD=188)
```

DD statement DD1 defines a new data set. The LABEL parameter tells the system:

- This data set is to be the third data set on the tape volume.
- This tape volume has nonstandard labels.
- This data set is to be kept for 188 days.

Although LABEL=(3,NSL,RETPD=188) is valid, it is better practice to use the DD RETPD parameter as follows:

```
//DD1   DD   DSNAME=HERBI,DISP=(NEW,KEEP),UNIT=TAPE,
//           VOLUME=SER=T2,LABEL=(3,NSL),RETPD=188
```

```
//DD2   DD   DSNAME=A.B.C,DISP=(,CATLG,DELETE),UNIT=3400-5,LABEL=(,NL)
```

DD statement DD2 defines a new data set, requests that the system catalog it, and indicates that the data set has no labels. Each time this data set is used by a program, the DD statement must include LABEL=(,NL).

```
//DD3   DD   DSNAME=SPECS,UNIT=3400-5,VOLUME=SER=10222,
//           DISP=OLD,LABEL=4
```

DD statement DD3 indicates an existing data set. The LABEL parameter indicates that the data set is fourth on the tape volume.

```
//STEP1  EXEC PGM=FIV
//DDX    DD   DSNAME=CLEAR,DISP=(OLD,PASS),UNIT=3400-5,
//            VOLUME=SER=1257,LABEL=(,NSL)
//STEP2  EXEC PGM=BOS
//DDY    DD   DSNAME=*.STEP1.DDX,DISP=OLD,LABEL=(,NSL)
```

DD statement DDX in STEP1 indicates an existing data set with nonstandard labels and requests that the system pass the data set. DD statement DDY in STEP2 receives the data set. DDY contains the label type, because the system does not obtain the label type through the backward reference in the DSNAME parameter.

```
//DDZ  DD  DSNAME=CATDS,DISP=OLD,LABEL=(,SUL)
```

DD statement DDZ indicates an existing, cataloged data set on direct access. The data set has IBM standard labels and user labels. The LABEL parameter is required; otherwise, if the DD statement does not contain a LABEL parameter, the system assumes that a direct access data set has SL labels.

```
//DD7  DD  DSNAME=TOM1,DISP=(NEW,KEEP),LABEL=EXPDT=2006/033,
//         UNIT=3350,SPACE=(TRK,(1,1)),VOLUME=SER=663344
```

DD statement DD7 defines a new data set, requests the system to keep the data set, and indicates that the data set cannot be deleted or written over until the expiration date of February 2, 2006.

Although LABEL=EXPDT=2006/033 is valid, it is better practice to use the DD EXPDT parameter as follows:

```
//DD7  DD  DSNAME=TOM1,DISP=(NEW,KEEP),EXPDT=2006/033,
//         UNIT=3350,SPACE=(TRK,(1,1)),VOLUME=SER=663344
```

LIKE Parameter

Parameter Type: Keyword, optional — use this parameter only with SMS

Without SMS, use the DCB=dsname form of the DCB parameter described on page 10-45.

Purpose: Use the LIKE parameter to specify the allocation attributes of a new data set by copying the attributes of a model data set, which must be an existing cataloged data set.

The following attributes are copied from the model data set to the new data set:

- Data set organization
  - Record organization (RECORG) or
  - Record format (RECFM)
- Record length (LRECL)
- Key length (KEYLEN)
- Key offset (KEYOFF)
- Space allocation (AVGREC and SPACE)

If SMS is not installed or is not active, the system syntax checks and then ignores the LIKE parameter.

The retention period (RETPD) or expiration date (EXPDT) is not copied to the new data set.

Note: Do not use the LIKE parameter to copy attributes from a temporary data set (&&dsname), partitioned data set if a member name is included, and relative generation number for a GDG.

Syntax:

```
LIKE=data-set-name
```

Subparameter Definition

data-set-name
Specifies the data set name (dsname) of the model data set whose attributes are to be used as the attributes of the new data set.

Overrides

Any attributes you specify on the same DD statement with the following parameters override the corresponding attributes obtained from the model data set.

RECORG (record organization) or RECFM (record format)
LRECL (record length)
KEYLEN (key length)
KEYOFF (key offset)
AVGREC (record request and space quantity)
SPACE (average record length, primary, secondary, and directory quantity)

DD: LIKE

Relationship to Other Parameters

Do not code the following DD parameters with the LIKE parameter.

DYNAM
REFDD
SYSOUT

Examples of the LIKE Parameter

```
//SMSDS6   DD  DSNAME=MYDS6.PGM,LIKE=MYDSCAT.PGM,DISP=(NEW,KEEP)
```

In the example, the data set attributes used for MYDS6.PGM are obtained from the cataloged model data set MYDSCAT.PGM.

```
//SMSDS7   DD  DSNAME=MYDS7.PGM,LIKE=MYDSCAT.PGM,DISP=(NEW,KEEP),
//             LRECL=1024
```

In the example, the data set attributes used for MYDS7.PGM are obtained from the cataloged model data set MYDSCAT.PGM. Also, the logical record length of 1024 overrides the logical record length obtained from the model data set.

LRECL Parameter

Parameter Type: Keyword, optional

Purpose: Use the LRECL parameter to specify the length of the records in a data set.

Code the LRECL parameter when you want to (1) specify the logical record length for the data set or (2) with SMS, override the record length defined in the data class of the data set.

LRECL applies to data sets with the BPAM, BSAM, EXCP, QISAM, QSAM, and TCAM access methods, and with SMS, to VSAM data sets.

Note: Other than information for SMS, this information for the LRECL parameter previously appeared under the DCB LRECL subparameter.

Syntax:

```
LRECL=bytes
```

Subparameter Definition

bytes
Specifies (1) the length, in bytes, for fixed length records or (2) the maximum length, in bytes, for variable-length records.

The value of bytes is:

- 1 to 32760 for non-VSAM data sets.
- 1 to 32761 for VSAM key-sequenced (KS), entry-sequenced (ES), or relative record (RR) data sets. (LRECL does not apply to VSAM linear space, RECORG=LS, data sets.)

  For VSAM key-sequenced (KS) data sets, a record length must be specified, either explicitly with the LRECL or LIKE parameter, or in the data class for the data set. The record length must be greater than the key length.

Note: When RECFM is F or U, the length must not exceed DCB BLKSIZE. For RECFM=D or V, the length must not exceed BLKSIZE minus 4. For RECFM=VS, the length can exceed BLKSIZE. For unblocked records when DCB RKP=0, the length is for only the data portion of the record.

Additional Syntax for LRECL=bytes

LRECL=nnnnnK
Specifies the length in kilobytes for variable-length spanned records in ISO/ANSI/FIPS Version 3 tape data sets that are processed by the Data Facility Product using the extended logical record interface (XLRI). **nnnnn** is from 1 through 16383 and indicates multiples of 1024 bytes. The value in the DCB must be LRECL=0K or LRECL=nnnnnK. If a **K** is coded for any other type of data set, only the numeric value of LRECL is recognized.

LRECL=X
For QSAM only, specifies that the logical record length exceeds 32760 bytes for variable-length spanned records. This option is not valid for ISO/ANSI/FIPS Version 3 variable-length records.

Overrides

LRECL overrides the record length specified in the data set label, and with SMS, LRECL overrides the record length defined in the data class for the data set.

Relationship to Other Parameters

Do not code the following DD parameters with the LRECL parameter.

DCB=LRECL
DDNAME
DYNAM

Examples of the LRECL Parameter

```
//DD1B     DD  DSNAME=EVER,DISP=(NEW,KEEP),UNIT=3380,
//             RECFM=FB,LRECL=326,SPACE=(23472,(200,40))
```

In the example, the logical record length of 326 is used for the new data set EVER.

```
//SMSDS2   DD  NAME=MYDS2.PGM,DATACLAS=DCLAS02,DISP=(NEW,KEEP),
//             LRECL=256
```

In the example, the logical record length of 256 overrides the logical record length defined in the data class for the data set.

MGMTCLAS Parameter

Parameter Type: Keyword, optional — use this parameter only with SMS and for SMS-managed data sets

Without SMS, there are no DD parameters that provide this function.

Purpose: Use the MGMTCLAS parameter to specify a management class for a new SMS-managed data set. The storage administrator at your installation defines the names of the management classes you can code on the MGMTCLAS parameter.

After the data set is allocated, attributes in the management class control:

- The migration of the data set, which includes migration from primary storage to DFHSM-owned storage to archival storage.
- The backup of the data set, which includes frequency of backup, number of versions, and retention criteria for backup versions.

The Hierarchical Storage Manager (DFHSM) or a functionally equivalent program performs these functions.

If SMS is not installed or is not active, the system syntax checks and then ignores the MGMTCLAS parameter.

SMS ignores the MGMTCLAS parameter if you specify it for an existing data set.

The use of a management class can be protected by RACF.

References: See *MVS/XA Interactive Storage Management Facility User's Guide* for information on how to use ISMF to view your installation-defined management classes.

Syntax:

```
MGMTCLAS=management-class-name
```

Subparameter Definition

management-class-name

Specifies the name of a management class to be used for management of the SMS-managed data set after the data set is allocated.

The name, one to eight characters, is defined by the storage administrator at your installation.

DD: MGMTCLAS

Defaults

If you do not specify MGMTCLAS for a new data set and the storage administrator has provided an installation-written automatic class selection (ACS) routine, the ACS routine may select a management class for the data set. Check with your storage administrator to determine if an ACS routine will select a management class for the new data set, in which case you do not need to specify MGMTCLAS.

Overrides

You cannot override management class attributes via JCL parameters.

The management class for a data set defines a maximum value for the expiration date or retention period of the data set. This maximum limits the values that are specified on the EXPDT or RETPD parameter, or defined in the data class for the data set.

An ACS routine can override the management class that you specify on the MGMTCLAS parameter.

Relationship to Other Parameters

Do not code the following DD parameters with the MGMTCLAS parameter.

*	DYNAM
DATA	QNAME
DDNAME	

Code MGMTCLAS only when you specify a storage class for the data set (via the STORCLAS parameter) or an ACS routine selects a storage class.

Example of the MGMTCLAS Parameter

```
//SMSDS1  DD  DSNAME=MYDS1.PGM,DATACLAS=DCLAS1,STORCLAS=SCLAS1,
//            MGMTCLAS=MCLAS01,DISP=(NEW,KEEP)
```

In the example, SMS uses the attributes in the management class named MCLAS01 to handle the migration and backup of the SMS-managed data set. Note that installation-written ACS routines may override the specified management class, storage class, and data class.

MODIFY Parameter

Parameter Type: Keyword, optional

Purpose: Use the MODIFY parameter to specify a copy-modification module that tells JES how to print this sysout data set on a 3800 Printing Subsystem. The module can specify the following:

- Legends.
- Column headings.
- Where and on which copies the data is to be printed.

The module is defined and stored in SYS1.IMAGELIB using the IEBIMAGE utility program.

Note: MODIFY applies only for the 3800 Printing Subsystem Model 1 and 2 and the 3800 Printing Subsystem Model 3, 6, and 8 in compatibility mode.

References: For more information on the copy modification module and the IEBIMAGE utility program, see *Data Administration: Utilities.*

Syntax:

```
MODIFY={module-name          }
       {(module-name[,trc])  }
```

- You must code the module-name.
- The trc subparameter is optional. If you omit it, you can omit the parentheses. However, if you omit it, you must not code it as a null; for example, MODIFY=(TAB1,) is invalid.

Subparameter Definition.

module-name
Identifies a copy-modification module in SYS1.IMAGELIB. The module-name is 1 through 4 alphanumeric or national ($, #, @) characters.

trc
Identifies which table-name in the CHARS parameter is to be used. This **table reference character** is 0 for the first table-name specified, 1 for the second, 2 for the third, or 3 for the fourth. The CHARS parameter is on the following, in override order:

1. This DD statement.
2. A referenced OUTPUT JCL statement.
3. A statement in the library member specified on the OUTPUT JCL PAGEDEF parameter.
4. A statement in the SYS1.IMAGELIB member obtained by default.
5. A JES3 initialization statement.

DD: MODIFY

Defaults

If no MODIFY parameter is specified, JES3 uses an installation default specified at initialization. JES2 provides no installation default at initialization.

If you do not specify **trc** or if the **trc** value is greater than the number of table-names in the CHARS parameter, JES2 uses the first table named in the CHARS parameter and JES3 uses the default character arrangement table.

Overrides

A MODIFY parameter on a sysout DD statement overrides an OUTPUT JCL MODIFY parameter.

Note: A null first subparameter is invalid in a MODIFY parameter on a DD statement, but is permitted on an OUTPUT JCL statement.

Relationship to Other Parameters

Do not code the following parameters with the MODIFY parameter.

*	DSID	PROTECT
AMP	DSNAME	QNAME
DATA	DYNAM	SUBSYS
DDNAME	LABEL	VOLUME
DISP	MSVGP	

Relationship to Other Control Statements

MODIFY can also be coded on the following:

- The OUTPUT JCL statement.
- The JES3 //*FORMAT PR statement.
- The JES2 /*OUTPUT statement.

The second character of each logical record can be a TRC code, so that each record can be printed in a different font. This way of specifying fonts is indicated by the OUTPUT JCL TRC parameter.

Example of the MODIFY Parameter

```
//DD1 DD  UNIT=3800,MODIFY=(A,0),CHARS=(GS15,GS10)
```

In this example, the MODIFY parameter requests that the data in the copy-modification module named A replace variable data in the data set to be printed by the 3800. Module A defines which positions are to be replaced and which copies are to be modified. The second subparameter in MODIFY specifies that the first character arrangement table in the CHARS parameter, GS15, be used.

MSVGP Parameter

Parameter Type: Keyword, optional

Note that SMS does not manage mass storage volumes for SMS-managed data sets.

Purpose: Use the MSVGP parameter to place the data set in a group of mass storage volumes on a mass storage system (MSS) device. MSVGP applies only to a nonspecific volume request, which is a DD statement for a new data set that can be assigned to any volume or volumes.

References: For information on defining mass storage groups, see *Mass Storage System (MSS) Services General Information.*

Syntax:

```
        ┌id              ┐
MSVGP=  │(id[,ddname])   │
        └SYSGROUP        ┘
```

You can omit the parentheses if you code only the id subparameter or SYSGROUP.

Subparameter Definition

id

Identifies a group of mass storage volumes. The id is 1 through 8 alphanumeric or national ($, #, @) characters and must be previously defined by the installation using a mass storage system service.

ddname

Requests that the data set for this DD statement is to be allocated to volume(s) other than the volume(s) occupied by the data set for DD statement ddname. DD statement ddname must appear earlier in the job step.

If volume separation within the group is not possible, the system terminates the job.

The ddname refers to only the first data set (1) when the named DD statement starts a concatenation or (2) when the DSNAME parameter of the named DD statement requests all members of a generation data group (GDG).

SYSGROUP

Identifies a default group of mass storage volumes. Code MVSGP=SYSGROUP to make sure that a nonspecific request is allocated to SYSGROUP.

DD: MSVGP

Relationship to Other Parameters

Do not code the following parameters with the MSVGP parameter.

*	DDNAME	QNAME
BURST	DSID	SYSOUT
CHARS	DYNAM	
COPIES	FLASH	
DATA	MODIFY	

SPACE Parameter: Code the SPACE parameter for the following reasons:

- To allocate noncontiguous primary space. Contiguous space is the MSVGP default.
- For nonspecific requests for BPAM and ISAM data sets.

Do not code the ABSTR, MXIG, and ALX subparameters in the SPACE parameter.

VOLUME Parameter: Do not code VOLUME=SER with the MSVGP parameter; VOLUME=SER is a specific volume request, while MSVGP can be used only for nonspecific volume requests.

VOLUME=PRIVATE with MSVGP is redundant, because MSVGP causes allocation to a private volume.

UNIT and VOLUME Counts: The unit count in the UNIT parameter must be less than the volume count in the VOLUME parameter to guarantee allocation of a unit that is not shared with another job.

Allocation when MSVGP is Not Coded

When a DD statement defines a new, nonspecific data set to be placed on an MSS device and does not contain an MSVGP parameter, the system allocates as follows:

- Places a permanent data set on a mounted 3330V Disk Storage volume, if one exists. If one does not exist, a volume is selected from SYSGROUP; in this case, the DD statement must contain a SPACE parameter or the job is terminated.

- Places a temporary data set on a mounted 3330V Disk Storage public volume, if one exists. If one does not exist or it does not have enough space, a volume is selected from SYSGROUP.

Examples of the MSVGP Parameter

```
//DD1 DD DSNAME=ACCOUNT,DISP=(NEW,CATLG),UNIT=3330V,
//       MSVGP=AB$1234@,VOLUME=(,,,3)
```

A new, cataloged data set is to be created on one, two, or three mass storage volumes in the group called AB$1234@. The installation previously defined this group and assigned at least three volumes to it. If the installation provided a space default of SPACE=(CYL,(200,100)), the system selects from the group a volume with at least 200 cylinders available.

During step execution, if more than 200 cylinders are required, and if 100 more cylinders are not available on the mounted volume, the system asks the operator to demount the volume. The system selects from group AB$1234@ a volume with at least 100 cylinders available and asks the operator to mount the volume. The volume count of three allows the data set to extend over three volumes. If more are required, the step abnormally terminates.

```
//DD1 DD DSNAME=MASTRIN,DISP=OLD
//DD2 DD DSNAME=MASTROUT,UNIT=3330V,DISP=(,CATLG),
//       MSVGP=(AB$1234@,DD1)
```

DD1 requests an existing cataloged data set. DD2 defines a new data set that will be allocated to a volume in mass storage group AB$1234@.

Because DD1 is specified as the ddname subparameter of MSVGP on DD2, the system allocates the DD2 data set, MASTROUT, to different volumes than the volumes for the DD1 data set, MASTRIN.

OUTLIM Parameter

Parameter Type: Keyword, optional

Purpose: Use the OUTLIM parameter to limit the number of logical records in the sysout data set defined by this DD statement. When the limit is reached, the system exits to the SYSOUT limit exit routine. If the installation supplies a user-written routine, the routine can determine whether to terminate the job or increase the limit. If the installation does not supply a routine, the system terminates the job.

If the installation supplies a routine, it must be in the SYS1.LPALIB library.

Note: OUTLIM is valid only on a DD statement with a SYSOUT parameter.

References: For more information on the SYSOUT limit exit routine, see *SPL: System Management Facilities.*

Syntax:

```
OUTLIM=number
```

Subparameter Definition

number
Specifies the maximum number of logical records. The number is 1 through 8 decimal digits from 1 through 16777215.

Default

(1) If no OUTLIM parameter is specified or OUTLIM=0 is coded and (2) if output is not limited by JES control statements, JES3 uses an installation default specified at initialization; JES2 provides no installation default at initialization.

Relationship to Other Parameters

Code the OUTLIM parameter only on a DD statement with the SYSOUT parameter.

Do not code the OUTLIM parameter with the DCB subparameters CPRI or THRESH; these subparameters can alter the OUTLIM value.

On Dump DD Statements: JES3 ignores an OUTLIM parameter on a SYSABEND or SYSUDUMP DD statement.

Relationship to Other Control Statements

Output can also be limited by the following:

- The LINES, BYTES, PAGES, or CARDS parameter of the JES2 /*JOBPARM statement.
- The LINES, BYTES, PAGES, or CARDS parameter of the JES3 //*MAIN statement.

Example of the OUTLIM Parameter

```
//OUTDD  DD  SYSOUT=F,OUTLIM=1000
```

The limit for the number of logical records is 1000.

OUTPUT Parameter

Parameter Type: Keyword, optional

Purpose: Use the OUTPUT parameter with the SYSOUT parameter to associate a sysout data set explicitly with an OUTPUT JCL statement. JES processes the sysout data set using the options from this DD statement combined with the options from the referenced OUTPUT JCL statement.

When the OUTPUT parameter references more than one OUTPUT JCL statement, the system produces separate output for each OUTPUT JCL statement.

Note: Code the OUTPUT parameter only on a DD statement with a SYSOUT parameter. Otherwise, the system checks the OUTPUT parameter for syntax then ignores it.

Syntax:

```
OUTPUT={reference                    }
       {(reference[,reference]...)   }

A reference is one of the following:

    *.name
    *.stepname.name
    *.stepname.procstepname.name
```

- You can omit the parentheses if you code only one reference.
- You must not code a null in an OUTPUT parameter. For example, OUTPUT=(,*.name) is invalid.
- You can reference a maximum of 128 OUTPUT JCL statements on one OUTPUT parameter.
- You can code references in any combination. For example, the following are valid:

```
//EXA  DD   SYSOUT=A,OUTPUT=(*.name,*.name,*.stepname.name)
//EXB  DD   SYSOUT=A,OUTPUT=(*.stepname.name,
//             *.stepname.procstepname.name,*.name)
```

- You can code the references to OUTPUT JCL statements in any order.

Subparameter Definition

***.name**
Refers to an earlier OUTPUT JCL statement with **name** in its name field. The system searches for the OUTPUT JCL statement first in the same step, then before the first EXEC statement of the job.

***.stepname.name**
Refers to an earlier OUTPUT JCL statement, **name**, in this step or an earlier step, stepname, in the same job.

***.stepname.procstepname.name**
Refers to an OUTPUT JCL statement in a cataloged or in-stream procedure. Stepname is the name of this job step or an earlier job step that calls the procedure, procstepname is the name of the procedure step that contains the OUTPUT JCL statement, and name is the name field of the OUTPUT JCL statement.

Defaults

If you do not code an OUTPUT parameter on a sysout DD statement, JES obtains processing options for the sysout data set in the following order:

1. From each OUTPUT JCL statement containing DEFAULT=YES in the same step.
2. From each OUTPUT JCL statement containing DEFAULT=YES before the first EXEC statement in the job, provided that the step contains no OUTPUT JCL statements with DEFAULT=YES.
3. Only from the sysout DD statement, provided that neither the step nor job contains any OUTPUT JCL statements with DEFAULT=YES.

Overrides

When an OUTPUT JCL statement is used with the sysout DD statement to specify processing, JES handles parameters as follows:

- If a parameter appears on the DD statement, JES uses the parameter.
- If a parameter appears only on the OUTPUT JCL statement, JES uses the parameter.
- If the same parameter appears on both statements, JES uses the DD parameter.

JES uses the whole overriding parameter, ignoring the whole overridden parameter. If a subparameter is left off the overriding parameter, the system does not pick up that subparameter from the overridden parameter. For example:

```
//EXAMP2   OUTPUT    FLASH=(ABCD,3)
//FVZ2     DD        SYSOUT=F,OUTPUT=*.EXAMP2,FLASH=(EFGH)
```

Only EFGH is used. The system ignores all of the FLASH parameter on the OUTPUT JCL statement, including the second parameter.

DD: OUTPUT

Relationship to Other Parameters

Code the OUTPUT parameter only on a DD statement with the SYSOUT parameter.

With INTRDR Subparameter in SYSOUT Parameter: Do not code an OUTPUT parameter when the SYSOUT parameter specifies a JES2 internal reader by an INTRDR parameter.

Null Subparameters: A null first subparameter is invalid in a FLASH or MODIFY parameter on a DD statement, but is permitted on an OUTPUT JCL statement. For example, MODIFY=(,3) is valid only on an OUTPUT JCL statement.

SYSOUT Third Subparameter: You cannot reference a JES2 /*OUTPUT statement using the third subparameter of the SYSOUT parameter if either of the following is also coded:

- The OUTPUT parameter on the same DD statement.
- An OUTPUT JCL statement containing DEFAULT=YES in the same step or before the EXEC statement of the job, when the DD statement does not contain an OUTPUT parameter.

DEFAULT Parameter on OUTPUT JCL Statement: If you code DEFAULT=YES on an OUTPUT JCL statement, you can still refer to that OUTPUT JCL statement in the OUTPUT parameter of a sysout DD statement.

Location in the JCL

All referenced OUTPUT JCL statements must precede the DD statement that refers to them. If the referencing DD statement appears in an in-stream or cataloged procedure, the referenced OUTPUT JCL statement must precede the DD statement in the procedure. A sysout DD statement in a procedure cannot refer to an OUTPUT JCL statement in the calling step.

No Match for OUTPUT Name

If the system finds no match for the name coded in the OUTPUT parameter, the system issues a JCL error message and fails the job.

Processing Options in Multiple References

A sysout DD statement can refer to more than one OUTPUT JCL statement, either explicitly in an OUTPUT parameter containing more than one reference or implicitly when several default OUTPUT JCL statements apply. The processing options for a sysout data set come from one sysout DD statement and one OUTPUT JCL statement. In multiple references, each combination of sysout DD statement and one of the referenced OUTPUT JCL statements produces a separate set of printed or punched output.

Processing options are **not** cumulative across a group of OUTPUT JCL statements.

Examples of the OUTPUT Parameter

```
//J1      JOB     ,'MARY LUDWIG'
//JOUT    OUTPUT CLASS=C,FORMS=RECP,INDEX=6
//STEP1   EXEC    PGM=XYZ
//SOUT    OUTPUT CLASS=H,BURST=YES,CHARS=GT12,FLASH=BLHD
//ALL     DD      SYSOUT=(,),OUTPUT=(*.JOUT,*.SOUT),COPIES=5
//IN      DD      *
          .
         (data)
          .
/*
```

The OUTPUT parameter references two OUTPUT JCL statements. Therefore, the system prints the single sysout data set twice:

- For DD ALL combined with OUTPUT JOUT, the sysout data set is printed in class C. In the installation, output class C is printed on a 3211 Printer. Combining the parameters from the DD and OUTPUT JCL statements, the system prints 5 copies of the data set on form RECP and indents the left margin 5 spaces.

- For DD ALL combined with OUTPUT SOUT, the sysout data set is printed in class H. In the installation, output class H is printed on a 3800 Printing Subsystem. Combining the parameters from the DD and OUTPUT JCL statements, the system prints 5 copies of the data set with the forms-overlay frame named BLHD using character-arrangement table GT12 and bursts the output.

```
//J6      JOB     ,'SUE THACKER'
//OUTA    OUTPUT DEST=HQ
//STEP1   EXEC    PGM=RDR
//OUTB    OUTPUT CONTROL=DOUBLE
//DS1     DD      SYSOUT=A,OUTPUT=(*.OUTA,*.OUTB)
//STEP2   EXEC    PGM=WRT
//OUTC    OUTPUT DEST=ID2742
//DS2     DD      SYSOUT=A,OUTPUT=(*.OUTC,*.STEP1.OUTB)
```

The OUTPUT parameter on DS1 references:

- The job-level OUTPUT JCL statement OUTA to send the sysout data set to HQ.

- The step-level OUTPUT JCL statement OUTB to print the sysout data set double-spaced on the local 3800 Printing Subsystem used for output class A.

The OUTPUT parameter on DS2 references:

- OUTPUT JCL statement OUTB in the first step to print the sysout data set double-spaced on the local 3800 Printing Subsystem used for output class A.

- OUTPUT JCL statement OUTC in the same step to send the sysout data set to userid ID2742, which is attached to the local system.

Note that the references to OUTPUT JCL statements are in no particular order.

PROTECT Parameter

Parameter Type: Keyword, optional

Use the PROTECT parameter only if RACF is installed and active.

With SMS, use the SECMODEL parameter to protect data sets; SECMODEL is described on page 10-147.

Purpose: Use the PROTECT parameter to tell the Resource Access Control Facility (RACF) to protect:

- One data set on a direct access volume.
- One data set on a tape volume with one of the following types of labels:
  - IBM standard labels, LABEL=(,SL) or LABEL=(,SUL)
  - ISO/ANSI/FIPS Version 3 labels, LABEL=(,AL) or LABEL=(,AUL)
  - Nonstandard labels, LABEL=(,NSL), if the installation provides support
- An entire tape volume with one of the following:
  - IBM standard labels, LABEL=(,SL) or LABEL=(,SUL)
  - ISO/ANSI/FIPS Version 3 labels, LABEL=(,AL) or LABEL=(,AUL)
  - Nonstandard labels, LABEL=(,NSL), if the installation provides support
  - No labels, LABEL=(,NL)
  - Bypassed label processing, LABEL=(,BLP)
  - Leading tapemarks, LABEL=(,LTM)

References: For more information on RACF, see *Resource Access Control Facility (RACF) General Information Manual.*

Syntax:

```
PROTECT={YES}
        {Y  }
```

Subparameter Definition

YES
Requests RACF to protect a direct access data set, tape data set, or tape volume. This parameter can also be coded as Y.

Overrides

With SMS, the DD SECMODEL parameter overrides the PROTECT=YES parameter.

Relationship to Other Parameters

Do not code the following parameters with the PROTECT parameter.

*	DLM	QNAME
BURST	DYNAM	SYSOUT
CHARS	FCB	TERM
DATA	FLASH	UCS
DDNAME	MODIFY	

DSNAME Parameter for RACF-Protected Data Sets: RACF expects the data set name specified in the DSNAME parameter to have a high-level qualifier that is defined to RACF. See the *RACF Security Administrator's Guide* for details.

Requirements for Protecting a Tape Data Set

A DD statement that contains a PROTECT parameter to establish RACF protection for a tape data set must:

- Specify or imply VOLUME=PRIVATE.
- Specify or imply DISP=NEW, DISP=OLD, or DISP=SHR; it must not specify or imply DISP=MOD.
- Specify in the LABEL parameter a label type of:
  - SL or SUL for IBM standard labels.
  - AL or AUL for ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 tape labels.
  - NSL for nonstandard labels. In this case, the NSL installation exit routine must issue a RACDEF or RACROUTE TYPE=DEFINE macro instruction. See *SPL: System Macros and Facilities* for a description of these macro instructions.
- Specify in the LABEL parameter a data-set-sequence-number, if the data set is not the first on the volume.

Requirements for Protecting a Tape Volume

A DD statement that contains a PROTECT parameter to establish RACF protection for a tape volume must:

- Specify or imply VOLUME=PRIVATE.
- Specify or imply DISP=NEW.
- Specify in the LABEL parameter a label type of:
  - SL or SUL for IBM standard labels.

- AL or AUL for ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 tape labels.
- NSL for nonstandard labels. In this case, the NSL installation exit routine must issue a RACDEF or RACROUTE TYPE=DEFINE macro instruction.
- NL for no labels.
- BLP for bypass label processing.
- LTM for leading tapemark.

Note that RACF cannot fully protect unlabeled tapes because RACF cannot verify the volume serial number directly; the operator must verify the volume serial number when mounting the tape volume.

Requirements for Protecting a Direct Access Data Set

A DD statement that contains a PROTECT parameter to establish RACF protection for a direct access data set must:

- Name a permanent data set in the DSNAME parameter.
- Specify a status of DISP=NEW or MOD treated as NEW. RACF can establish protection only when the data set is being created.

Examples of the PROTECT Parameter

```
//DASD  DD  DSNAME=USER37.MYDATA,DISP=(,CATLG),
//      VOLUME=SER=333000,UNIT=3330,SPACE=(TRK,2),PROTECT=YES
```

This DD statement requests RACF protection for the new direct access data set USER37.MYDATA.

```
//TAPEVOL  DD  DSNAME=MHB1.TAPEDS,DISP=(NEW,KEEP),LABEL=(,NL),
//           VOLUME=SER=T49850,UNIT=3400-5,PROTECT=YES
```

This DD statement requests RACF protection for tape volume T49850. Because a specific tape volume is requested, it automatically has the PRIVATE attribute. The volume has no labels.

```
//TAPEDS  DD  DSNAME=INST7.NEWDS,DISP=(NEW,CATLG),LABEL=(2,SUL),
//          VOLUME=SER=223344,UNIT=3400-5,PROTECT=YES
```

This DD statement requests RACF protection for INST7.NEWDS, which is the second data set on tape volume 223344. Because a specific tape volume is requested, it automatically has the PRIVATE attribute. The volume has IBM standard and user labels; the RACF Release 1.7 TAPEDSN facility must be active.

QNAME Parameter

Parameter Type: Keyword, optional

Purpose: Use the QNAME parameter to indicate that this DD statement defines a data set of telecommunications access method (TCAM) messages. The QNAME parameter refers to a TPROCESS macro instruction that defines a destination queue for the messages. Optionally, the QNAME parameter can also name a TCAM job to process the messages.

References: For information about TCAM and the TPROCESS macro instruction, see *Advanced Communications Function for TCAM, Version 2, Installation Reference.*

Syntax:

```
QNAME=procname[.tcamname]
```

Subparameter Definition

procname
Identifies a TPROCESS macro instruction; procname must be identical to the procname in the name field of the TPROCESS macro instruction.

tcamname
Names a TCAM job: tcamname must be identical to the jobname. The TCAM job can be a task started by an operator START command.

Relationship to Other Parameters

The only DD parameters that you can code with the QNAME parameter are DCB, LIKE, LRECL, RECFM, and REFDD. The only DCB subparameters that you can code with the QNAME parameter are: BLKSIZE, BUFL, LRECL, OPTCD, and RECFM.

Examples of the QNAME Parameter

```
//DYD   DD   QNAME=FIRST,DCB=(RECFM=FB,LRECL=80,BLKSIZE=320)
```

This DD statement defines a data set of TCAM messages. **FIRST** is the name of the TPROCESS macro instruction that specifies the destination queue to which the messages are routed. The DCB parameter supplies information not supplied in the program's DCB macro instruction for the data control block.

```
//DXD   DD    QNAME=SECOND.TCAM01
```

This DD statement defines a data set of TCAM messages. **SECOND** is the name of the TPROCESS macro instruction that specifies the destination queue to which the messages are routed. TCAM program TCAM01 will process the messages.

RECFM Parameter

Parameter Type: Keyword, optional

Purpose: Use the RECFM parameter to specify the format and characteristics of the records in a data set. All the format and characteristics must be completely described in one source, that is, in the data set label of an existing data set, in the DCB macro, in the DD DCB parameter, or in the DD RECFM parameter. However, the processing program can modify the RECFM field in the DCB.

Code the RECFM parameter when you want to (1) specify the record format for the data set or (2) with SMS, override the record format defined in the data class of the data set.

The syntax of the RECFM parameter is described in the following topics:

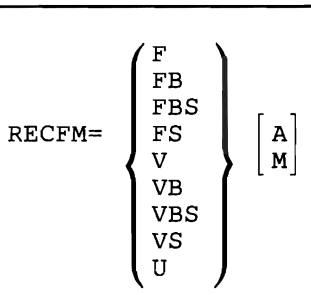
- Coding RECFM for Data Sets with SMS
- Coding RECFM for BDAM Access Method *
- Coding RECFM for BPAM Access Method *
- Coding RECFM for BSAM, EXCP, and QSAM Access Methods *
- Coding RECFM for QISAM Access Method *
- Coding RECFM for TCAM Access Method *

* These descriptions previously appeared under the DCB RECFM subparameter.

When indicated in the syntax diagrams, combinations of characteristics can be coded. For example, for data sets with SMS, FBS indicates that the records are fixed, blocked, and spanned.

Coding RECFM for Data Sets with SMS

Syntax: Data Sets with SMS



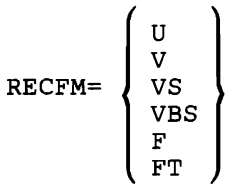
A or M can be coded with any record format, such as: RECFM = FBA

You can specify spanned (S) records only for physical sequential (PS) data sets.

B	indicates that the records are blocked.
F	indicates that the records are fixed length.
S	indicates that the records are spanned.
V	indicates that the records are variable length.
U	indicates that the records are undefined length.
A	indicates that the records contain ISO/ANSI control characters.
M	indicates that the records contain machine code control characters.

Coding RECFM for BDAM Access Method

Syntax: BDAM Access Method

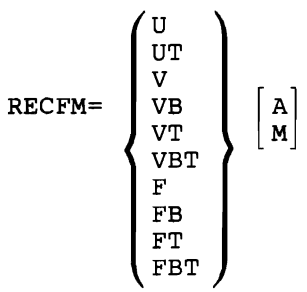


U	indicates that the records are undefined length.
V	indicates that the records are variable length.
S	indicates that the records are spanned.
F	indicates that the records are fixed length.
T	indicates that the records may be written using the track-overflow feature.

Default: undefined-length, unblocked records.

Coding RECFM for BPAM Access Method

Syntax: BPAM Access Method



A or M can be coded with any record format, such as: RECFM = FBA

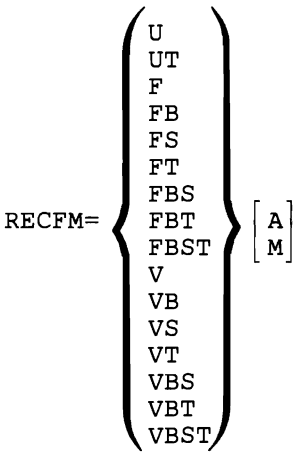
B	indicates that the records are blocked.
F	indicates that the records are fixed length.
T	indicates that the records may be written using the track-overflow feature. Chained scheduling (OPTCD = C) will be ignored.
U	indicates that the records are undefined length.
V	indicates that the records are variable length.
A	indicates that the records contain ISO/ANSI control characters.
M	indicates that the records contain machine code control characters.

Default: U

DD: RECFM

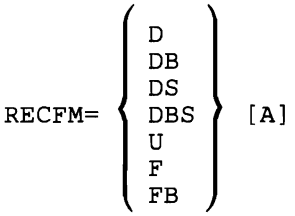
Coding RECFM for BSAM, EXCP, and QSAM Access Methods

Syntax: BSAM, EXCP, and QSAM Access Methods



A or M can be coded with any record format, such as: RECFM=FBA

For BSAM, EXCP, and QSAM using ISO/ANSI/FIPS data sets on tape:



A can be coded with any record format, such as: RECFM=FBA

A or M cannot be specified if the PRTSP subparameter is specified.

B indicates that the records are blocked.
D indicates that the records are variable-length ISO/ANSI tape records.
F indicates that the records are fixed length.
S (1) For fixed-length records, indicates that the records are written as standard blocks, that is, no truncated blocks or unfilled tracks within the data set, with the exception of the last block or track. (2) For variable-length records, indicates that a record can span more than one block.
T indicates that the records can be written using the track-overflow feature, if required. Chained scheduling (OPTCD=C) is ignored.
U indicates that the records are undefined length. U is invalid for an ISO/ANSI/FIPS Version 3 tape data set.
V indicates that the records are variable length. V cannot be specified for (1) a variable-length ISO/ANSI tape data set (specify D for this data set), (2) a card reader data set, or (3) a 7-track tape unless the data conversion feature (TRTCH=C) is used.
A indicates that the record contains ISO/ANSI device control characters.
M indicates that the records contain machine code control characters.

Default: U for IBM standard label tapes.

Coding RECFM for QISAM Access Method

Syntax: QISAM Access Method

```
RECFM= { V  }
       { VB }
       { F  }
       { FB }
```

B indicates that the records are blocked.
F indicates that the records are fixed length.
V indicates that the records are variable length; variable records cannot be in ASCII.

When creating indexed sequential data sets, you can code the RECFM subparameter; when processing existing indexed sequential data sets, you must omit RECFM.

Default: V

Coding RECFM for TCAM Access Method

Syntax: TCAM Access Method

```
RECFM= { U  }
       { V  }
       { VB }
       { F  }
```

B indicates that the records are blocked.
F indicates that the records are fixed length.
U indicates that the records are undefined length.
V indicates that the records are variable length.

Default: U

Overrides

RECFM overrides the record format specified in the data set label, and with SMS, RECFM overrides the record format defined in the data class for the data set.

Relationship to Other Parameters

Do not code the following DD parameters with the RECFM parameter.

*	DDNAME
AMP	DYNAM
DATA	RECORG
DCB=DSORG	
DCB=RECFM	

Examples of the RECFM Parameter

```
//DD1B  DD    DSNAME=EVER,DISP=(NEW,KEEP),UNIT=3380,
//            RECFM=FB,LRECL=326,SPACE=(23472,(200,40))
```

In the example, the record format of fixed block (FB) is used for the new data set EVER.

```
//SMSDS6  DD  DSNAME=MYDS6.PGM,DATACLAS=DCLAS06,DISP=(NEW,KEEP),
//              RECFM=FB
```

In the example, the record format of fixed block (FB) overrides the record format defined in the data class for the data set.

RECORG Parameter

Parameter Type: Keyword, optional — use this parameter only with SMS

Without SMS, see the AMP parameter described on page 10-18.

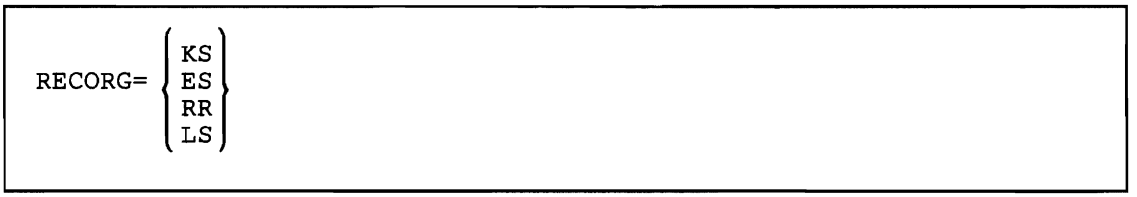
Purpose: Use the RECORG parameter to specify the organization of the records in a new VSAM data set.

Code the RECORG parameter when you want to (1) specify the record organization for the data set or (2) override the record organization defined in the data class of the data set.

If SMS is not installed or is not active, the system syntax checks and then ignores the RECORG parameter.

References: See *MVS/Extended Architecture VSAM Administration Guide* for information on VSAM data sets.

Syntax:



Subparameter Definition

KS
Specifies a VSAM key-sequenced data set.

ES
Specifies a VSAM entry-sequenced data set.

RR
Specifies a VSAM relative record data set.

LS
Specifies a VSAM linear space data set.

Defaults

If you do not specify RECORG, SMS assumes a physical sequential (PS) or partitioned (PO) data set.

DD: RECORG

Overrides

The RECORG parameter overrides the record organization defined in the data class for the data set.

Relationship to Other Parameters

Do not code the following DD parameters with the RECORG parameter.

*	DDNAME
DATA	DYNAM
DCB=DSORG	RECFM
DCB=RECFM	

Example of the RECORG parameter

```
//SMSDS3  DD  DSNAME=MYDS3.PGM,DATACLAS=VSAM1,DISP=(NEW,KEEP),
//            RECORG=KS
```

In the example, the record organization of key-sequenced (KS) overrides the record organization defined in the data class.

REFDD Parameter

Parameter Type: Keyword, optional — use this parameter only with SMS

Without SMS, use the DCB=*.ddname form of the DCB parameter described on page 10-46.

Purpose: Use the REFDD parameter to specify the attributes for a new data set by copying the attributes of a data set that is defined on an earlier DD statement in the same job.

The following attributes are copied to the new data set from (1) the attributes specified on the referenced DD statement, and (2) for attributes not specified on the referenced DD statement, from the data class of the referenced DD statement:

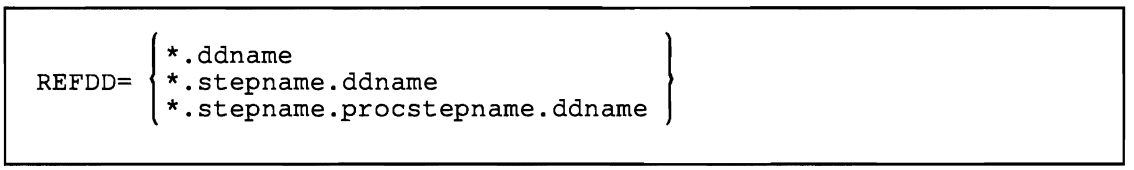
- Data set organization
  - Record organization (RECORG) or
  - Record format (RECFM)
- Record length (LRECL)
- Key length (KEYLEN)
- Key offset (KEYOFF)
- Space allocation (AVGREC and SPACE)

If SMS is not installed or is not active, the system syntax checks and then ignores the REFDD parameter.

The retention period (RETPD) or expiration date (EXPDT) is not copied to the new data set.

Note: Do not use the REFDD parameter to copy attributes from a temporary data set (&&dsname), partitioned data set if a member name is included, and relative generation number for a GDG.

Syntax:



Subparameter Definition

***.ddname**
***.stepname.ddname**
***.stepname.procstepname.ddname**
Specify a backward reference to an earlier DD statement. The referenced DD statement cannot name a cataloged data set or refer to another DD statement.

***.ddname**
Specifies the ddname of an earlier DD statement in the same step.

***.stepname.ddname**
Specifies the ddname of a DD statement in an earlier step, stepname, in the same job.

DD: REFDD

***.stepname.procstepname.ddname**
Specifies the ddname of a DD statement in a cataloged or in-stream procedure called by an earlier job step. Stepname is the name of the job step that calls the procedure and procstepname is the name of the procedure step that contains the DD statement.

Do not reference a DD * or a DD DATA statement.

Overrides

Any attributes specified on the referenced DD statement override the corresponding data class attributes of the referenced data set.

Any attributes you specify on the referencing DD statement with the following parameters override the corresponding attributes obtained from the referenced DD statement and the data class attributes of the referenced data set.

RECORG (record organization) or RECFM (record format)
LRECL (record length)
KEYLEN (key length)
KEYOFF (key offset)
AVGREC (record request and space quantity)
SPACE (average record length, primary, secondary, and directory quantity)

Relationship to Other Parameters

Do not code the following DD parameters with the REFDD parameter.

DYNAM
LIKE

Examples of the REFDD Parameter

```
//SMSDS6  DD  DSNAME=MYDS6.PGM,DATACLAS=DCLAS01,DISP=(NEW,KEEP),
//            LRECL=512,RECFM=FB
//SMSDS7  DD  DSNAME=MYDS7.PGM,REFDD=*.SMSDS6,DISP=(NEW,KEEP)
```

In the example, the data set attributes used for MYDS7.PGM are obtained from the referenced data set MYDS6.PGM.

```
//SMSDS6  DD  DSNAME=MYDS6.PGM,DATACLAS=DCLAS01,DISP=(NEW,KEEP),
//            LRECL=512,RECFM=FB
//SMSDS8  DD  DSNAME=MYDS8.PGM,REFDD=*.SMSDS6,DISP=(NEW,KEEP),
//            LRECL=1024
```

In the example, the data set attributes used for MYDS8.PGM are obtained from the referenced data set MYDS6.PGM. Also, the logical record length of 1024 overrides the logical record length obtained from the referenced data set.

RETPD Parameter

Parameter Type: Keyword, optional

Purpose: Use the RETPD parameter to specify the retention period for a new data set. After the retention period, the data set can be deleted or written over by another data set.

If the DD statement contains DISP=(NEW,DELETE) or the DISP parameter is omitted to default to NEW and DELETE, the system deletes the data set when the step terminates normally or abnormally, even though a retention period is also specified.

Do not specify RETPD for a temporary data set.

The RETPD parameter achieves the same result as the EXPDT parameter.

Code the RETPD parameter when you want to (1) specify a retention period for the data set or (2) with SMS, override the retention period defined in the data class for the data set.

Note: Other than information for SMS, this information for the RETPD parameter previously appeared under the LABEL RETPD subparameter.

Syntax:

```
RETPD=nnnn
```

Subparameter Definition

nnnn

Specifies the retention period, in days, for the data set. The nnnn is one through four decimal digits (0 - 9999).

The system adds nnnn to the current date to produce an expiration date. The calculated expiration date uses 365-day years and 366-day leap years.

Note: If you code RETPD and the calculated expiration date is December 31, 1999, the expiration date is set to January 1, 2000.

Overrides

With SMS, RETPD overrides the retention period defined in the data class for the data set.

With SMS, both the retention period specified on RETPD and defined in the data class for an SMS-managed data set can be limited by a maximum retention period defined in the management class for the data set.

DD: RETPD

Relationship to Other Parameters

Do not code the following DD parameters with the RETPD parameter.

*	DYNAM
DATA	EXPDT
DDNAME	SYSOUT

Deleting a Data Set Before its Retention Period Passes

To delete a data set before the retention period has passed, use one of the following:

- For data sets cataloged in a VSAM or integrated catalog facility catalog, use the DELETE command, as described in *Integrated Catalog Administration: Access Method Services Reference* or *VSAM Catalog Administration: Access Method Services Reference.*
- For data sets not cataloged in a VSAM or integrated catalog facility catalog, use the IEHPROGM utility, as described in *Data Administration: Utilities.*
- For the data set control block (DSCB), use the SCRATCH macro with the OVRD parameter, as described in *System-Data Administration.* Deletion of the DSCB makes the space occupied by the data set available for reallocation.

Examples of the RETPD Parameter

```
//DD1  DD  DSNAME=HERBI,DISP=(NEW,KEEP),UNIT=TAPE,
//         VOLUME=SER=T2,LABEL=(3,NSL),RETPD=188
```

In the example, the data set is not eligible for being deleted or written over for 188 days.

```
//SMSDS2  DD  DSNAME=MYDS2.PGM,DATACLAS=DCLAS02,DISP=(NEW,KEEP),
//            RETPD=732
```

In the example, the retention period of 732 days overrides the retention period defined in the data class for the data set.

SECMODEL Parameter

Parameter Type: Keyword, optional — use this parameter only with SMS

Without SMS, use the DD PROTECT parameter described on page 10-132.

Purpose: Use the SECMODEL parameter to specify the name of an existing Resource Access Control Facility (RACF) data set profile that is copied to the discrete data set profile that RACF builds for the new data set.

The following information from the RACF data set profile, which RACF uses to control access to the data set, is copied to the discrete data set profile of the new data set:

- OWNER - indicates the user or group assigned as the owner of the data set profile.
- ID - indicates the access list of users or groups authorized to access the data set.
- UACC - indicates the universal access authority associated with the data set.
- AUDIT/GLOBALAUDIT - indicates which access attempts are logged.
- ERASE - indicates that the data set is to be erased when it is deleted (scratched).
- LEVEL - indicates the installation-defined level indicator.
- DATA - indicates installation-defined information.
- WARNING - indicates that an unauthorized access causes RACF to issue a warning message but allow access to the data set.
- SECLEVEL - indicates the name of an installation-defined security level.

Use the SECMODEL parameter (1) when you want a different RACF data set profile than the default profile selected by RACF or (2) when there is no default profile.

If SMS is not installed or is not active, the system syntax checks and then ignores the SECMODEL parameter.

References: For information about RACF, see *MVS Resource Access Control Facility (RACF) Command Language Reference*.

Syntax:

```
SECMODEL=(profile-name[,GENERIC])
```

DD: SECMODEL

Subparameter Definition

profile-name
Specifies the name of a RACF model profile, discrete data set profile, or generic data set profile. The named profile is copied to the discrete data set profile of the new data set.

If a generic data set profile is named, GENERIC must also be coded.

GENERIC
Identifies that the profile-name refers to a generic data set profile.

Overrides

The SECMODEL parameter overrides the PROTECT=YES parameter.

Relationship to Other Parameters

Do not code the following DD parameters with the SECMODEL parameter.

*	DDNAME
DATA	DYNAM

Examples of the SECMODEL Parameter

```
//SMSDS4   DD  DSNAME=MYDS4.PGM,SECMODEL=(GROUP4.DEPT1.DATA),
//             DISP=(NEW,KEEP)
```

In the example, RACF uses the previously defined model data set profile named GROUP4.DEPT1.DATA to control access to the new data set.

```
//SMSDS5   DD  DSNAME=MYDS5.PGM,SECMODEL=(GROUP5.*,GENERIC),
//             DISP=(NEW,KEEP)
```

In the example, RACF uses the previously defined generic data set profile named GROUP5.* to control access to the new data set.

SPACE Parameter

Parameter Type: Keyword, optional

Note: With SMS, code the SPACE parameter when you want to (1) request space for a new data set or (2) override the space allocation defined in the data class for the data set. See the DATACLAS parameter (described on page 10-42) and the AVGREC parameter (described on page 10-24).

Purpose: Use the SPACE parameter to request space for a new data set on a direct access volume. You can request space in two ways:

- Tell the system how much space you want and let the system assign specific tracks.
- Tell the system the specific tracks to be allocated to the data set.

Letting the system assign the specific tracks is most frequently used. You specify only how space is to be measured — in tracks, cylinders, blocks, or records — and how many of those tracks, cylinders, blocks, or records are required.

The SPACE parameter has no meaning for tape volumes; however, if you assign a data set to a device class that contains both direct access devices and tape devices, for example, UNIT=SYSSQ, you should code the SPACE parameter.

Syntax:

For system assignment of space:

```
SPACE=((TRK,|CYL,|blklgth,|reclgth,)(primary-qty[,second-qty[,directory|,index]])[,RLSE][,CONTIG|,MXIG|,ALX|,][,ROUND])
```

To request specific tracks:

```
SPACE=(ABSTR,(primary-qty,address[,directory|,index]))
```

To request only directory space:

```
SPACE=(,(,,directory))
```

- You can omit the parentheses around the primary quantity if you do not code secondary, directory, or index quantities. For example, SPACE=(TRK,20,RLSE,CONTIG) or SPACE=(TRK,20). Note that, if you omit these inner parentheses, you also omit the commas within them.

- All the subparameters are positional. Code a comma to indicate an omitted subparameter if any others follow. Thus:
  - If you code primary and directory or index quantities and omit a secondary quantity, code a comma inside the inner parentheses to indicate the omission. For example, SPACE=(TRK,(20,,2)).
  - If you omit RLSE but code a following subparameter, code a comma to indicate the omission. For example, SPACE=(TRK,(20,10),,CONTIG) or SPACE=(TRK,20,,CONTIG).
  - If you omit CONTIG, MXIG, or ALX and ROUND follows, code a comma to indicate the omission. For example, SPACE=(400,30,RLSE,,ROUND). If RLSE is also omitted, this example becomes SPACE=(400,30,,,ROUND).

Subparameter Definition

System Assignment of Space

TRK
Requests that space be allocated in tracks.

CYL
Requests that space be allocated in cylinders.

blklgth — (only if AVGREC is not coded)
Specifies the average block length, in bytes, of the data. The system computes how many tracks to allocate. The blklgth is a decimal number from 1 through 65535.

reclgth — (only if AVGREC is coded)
With SMS, specifies the average record length, in bytes, of the data. The system computes the block size and how many tracks to allocate. The reclgth is a decimal number from 1 through 65535.

primary-qty
Specifies one of the following:

- For TRK, the number of tracks to be allocated.
- For CYL, the number of cylinders to be allocated.
- For a block length, the number of data blocks in the data set.
- For a record length, the number of records in the new data set. Use the AVGREC parameter to specify that the primary quantity represents units, thousands, or millions of records.

Note: When you specify TRK or CYL for a partitioned data set, the primary quantity includes the space for the directory. When you specify a block length or record length for a partitioned data set, the primary quantity does not include the directory space; the system assigns the directory to space outside the primary space assignment.

One volume must have enough available space for the primary quantity. If you request a particular volume and it does not have enough space available for your request, the system terminates the job step. Allow for track overflow when computing track requirements.

To request an entire volume, specify in the primary quantity the number of tracks or cylinders on the volume minus the number used by the volume table of contents (VTOC). The volume must not contain other data sets.

second-qty

Specifies the number of additional tracks, cylinders, blocks, or records to be allocated, if more space is needed. The system does not allocate additional space until it is needed.

With SMS, use the AVGREC parameter to specify that the secondary quantity represents units, thousands, or millions of records.

If the first subparameter specifies the average block length, the system computes the number of tracks for the secondary quantity from the second-qty number multiplied by one of the following, in order:

1. The SPACE average block length subparameter.
2. The block length in the BLKSIZE field of the data control block.

When you specify a secondary quantity and the data set requires additional space, the system allocates the specified quantity:

1. In contiguous tracks or cylinders, if available.
2. If not, in up to five extents.

The system can allocate up to 16 extents for a data set on a volume. An extent is space that may or may not be contiguous to other space allocated to the data set. The extents for a data set include the primary quantity space and user-label space.

Note: BDAM data sets cannot be extended.

When your program has filled a sequential data set's allocated space on a volume, the system determines where the following data is written as follows:

- If the disposition of the data set is NEW or MOD and the limit on the number of extents on a volume has not been reached, the system attempts to allocate the secondary quantity on the same volume.

- If the disposition of the data set is OLD or SHARE, the system examines the next volume specified for the data set.

  - If space has been allocated on the next volume for the data set, the next volume is used for the data set.

  - If space has not been allocated on the next volume for the data set, secondary space is allocated on the next volume for the data set.

  - If there is not another volume specified for the data set, the system attempts to allocate the secondary quantity on the current volume.

Note that your program should not write with a disposition of DISP=SHR unless you take precautions to prevent other programs from writing at the same time.

If the requested volumes have no more available space and if at least one volume is demountable, the system asks the operator to mount scratch (nonspecific) volumes until the secondary allocation is complete. If none of the volumes are demountable, the system abnormally terminates the job step.

directory
Specifies the number of 256-byte records needed in the directory of a partitioned data set.

Note: When creating a partitioned data set, you must request space for a directory.

With SMS, you can specify the number of directory records on the SPACE parameter without specifying any other subparameters. For example:

```
//DD12  DD   DSNAME=PDS.EXMP,DATACLAS=DCLAS12,SPACE=(,(,,20)),
//           DISP=(NEW,KEEP)
```

specifies 20 directory records for the data set. In this example, the number of specified directory records (20) overrides the number of directory records defined in the data class of the data set. (SMS uses all other space allocation attributes defined in the data class of the data set.)

index
For the index of an indexed sequential data set, specifies one of the following:

- For TRK, the number of tracks needed. The number of tracks must equal one or more cylinders.
- For CYL, the number of cylinders needed.

RLSE
Requests that space allocated to an output data set, but not used, is to be released when the data set is closed. Unused space is released only if the data set is open for output and the last operation was a write.

If you specify RLSE and an abnormal termination occurs, the system does not release unused space even though the data set is open.

Do not code the RLSE subparameter for an indexed sequential data set.

Coding RLSE for primary allocation does not prohibit use of secondary allocation. The secondary request for space is still in effect.

The system ignores a request to release unused space when a data set is closed if:

- Another job is sharing the data set.
- Another task in the same job is processing an OPEN, CLOSE, EOV, or FEOV request for the data set.
- Another data control block is open for the data set.

The RLSE subparameter is ignored when TYPE=T is coded in the CLOSE macro instruction.

When coding RLSE for an existing data set, code the unit of measurement and primary quantity as they appeared in the original request. For example, if the original request was:

```
SPACE=(TRK,(100,50))
```

you can release unused tracks when you retrieve the data set by coding:

```
SPACE=(TRK,(100),RLSE)
```

CONTIG
Requests that space allocated to the data set must be contiguous. This subparameter affects only primary space allocation.

If CONTIG is specified and contiguous space is not available, the system terminates the job step.

MXIG
Requests that space allocated to the data set must be (1) the largest area of available contiguous space on the volume and (2) equal to or greater than the primary quantity. This subparameter affects only primary space allocation.

Note: Do not code a MXIG subparameter for an indexed sequential data set.

ALX
Requests that up to five separate areas of contiguous space are to be allocated to the data set and each area must be equal to or greater than the primary quantity. This subparameter affects only primary space allocation.

Note: Do not code an ALX subparameter for an indexed sequential data set.

ROUND
When the first subparameter specifies the average block length, requests that space allocated to the data set must be equal to an integral number of cylinders. If the first subparameter specifies TRK, CYL, or reclgth, the system ignores ROUND.

Request for Specific Tracks

For an SMS-managed data set (one with an assigned storage class), do not code ABSTR.

ABSTR
Requests that the data set be allocated at the specified location on the volume.

primary-qty
Specifies the number of tracks to be allocated to the data set.

The volume must have enough available space for the primary quantity. If it does not, the system terminates the job step.

address
Specifies the track number of the first track to be allocated. Count the first track of the first cylinder on the volume as 0. Count through the tracks on each cylinder until you reach the track on which you want the data set to start.

Note: Do not request track 0.

directory
Specifies the number of 256-byte records needed in the directory of a partitioned data set.

Note: When creating a partitioned data set, you must request space for a directory.

index
Specifies the number of tracks needed for the index of an indexed sequential data set. The number of tracks must equal one or more cylinders.

Relationship to Other Parameters

Do not code the following parameters with the SPACE parameter.

*	DYNAM
DATA	QNAME
DDNAME	SUBSYS

With KEYLEN for Block Requests: If space is requested in blocks and the blocks have keys, code the DD parameter KEYLEN (or the DCB subparameter KEYLEN) on the DD statement and specify the key length.

SPACE for New Data Sets with SMS

With SMS, code the SPACE parameter with or without the AVGREC parameter when you want to (1) request space for the data set or (2) override the space allocation attributes defined in the data class for the data set.

SPACE for New Data Sets on Mass Storage Volumes

- The SPACE parameter must be coded for new data sets on mass storage volumes when VOLUME=SER is coded. It is optional when MSVGP is coded. If you code neither VOLUME=SER nor MSVGP, SPACE must be coded even if you code VOLUME=PRIVATE.

- Contiguous space is the MSVGP default. If you want a noncontiguous primary space allocation, you must specify the SPACE parameter.

Examples of the SPACE Parameter

```
//DD1  DD  DSNAME=&&TEMP,UNIT=MIXED,SPACE=(CYL,10)
```

The DD statement defines a temporary data set. The UNIT parameter requests any available tape or direct access volume; MIXED is the installation's name for a group of tape and direct access devices. If a tape volume is assigned, the SPACE parameter is ignored; if a direct access volume is assigned, the SPACE parameter is used to allocate space to the data set. The SPACE parameter specifies only the required subparameters: the type of allocation and a primary quantity. It requests that the system allocate 10 cylinders.

```
//DD2  DD  DSNAME=PDS12,DISP=(,KEEP),UNIT=3350,
//         VOLUME=SER=25143,SPACE=(CYL,(10,,10),,CONTIG)
```

The DD statement defines a new partitioned data set. The system allocates 10 cylinders to the data set, of which ten 256-byte records are for a directory. Since the CONTIG subparameter is coded, the system allocates 10 contiguous cylinders on the volume.

```
//REQUEST1 DD  DSNAME=EXM,DISP=NEW,UNIT=3330,VOLUME=SER=606674,
//          SPACE=(1024,75),DCB=KEYLEN=8
//REQUESTA DD  DSNAME=EXQ,DISP=NEW,UNIT=3380,
//          SPACE=(1024,75),DCB=KEYLEN=8
```

These DD statements request space in block lengths. The average block length of the data is 1024 bytes. 75 blocks of data are expected as output. Each block is preceded by a key eight bytes long. The system computes how many tracks are needed, depending on the device requested in the UNIT parameter.

```
//REQUEST2 DD  DSNAME=PET,DISP=NEW,UNIT=3330,VOLUME=SER=606674,
//          SPACE=(ABSTR,(5,1))
```

In this example, the SPACE parameter asks the system to allocate 5 tracks, beginning on the second track of the volume.

```
//DD3    DD  DSNAME=MULTIVOL,UNIT=3350,DISP=(,CATLG),
//           VOLUME=SER=(223344,223345),SPACE=(CYL,(554,554))
```

This example shows how to create a multivolume data set on two complete volumes. The two volumes do not contain other data sets. A volume on 3350 Direct Access Storage contains 555 cylinders. The unrequested cylinder contains the volume table of contents (VTOC).

```
//SMSDS3  DD  DSNAME=MYDS3.PGM,DATACLAS=DCLAS03,DISP=(NEW,KEEP),
//          SPACE=(128,(5,2)),AVGREC=K
```

In this example, the space allocation defined in the DCLAS03 data class is overridden by the SPACE and AVGREC parameters, which indicate an average record length of 128 bytes, a primary quantity of 5K (5,120) records, and a secondary quantity of 2K (2,048) records.

STORCLAS Parameter

Parameter Type: Keyword, optional — use this parameter only with SMS and for SMS-managed data sets

Without SMS or for non-SMS-managed data sets, use the UNIT parameter (described on page 10-173) and the VOLUME parameter (described on page 10-178).

Purpose: Use the STORCLAS parameter to specify a storage class for a new SMS-managed data set. The storage administrator at your installation defines the names of the storage classes you can code on the STORCLAS parameter.

The storage class contains the attributes that identify a storage service level to be used by SMS for storage of the data set. It replaces the storage attributes that are specified on the UNIT and VOLUME parameters for non-SMS-managed data sets.

An **SMS-managed data set** is defined as a data set that has a storage class assigned. A storage class is assigned when either (1) you specify the STORCLAS parameter or (2) an installation-written automatic class selection (ACS) routine selects a storage class for a new data set.

If SMS is not installed or is not active, the system syntax checks and then ignores the STORCLAS parameter.

SMS ignores the STORCLAS parameter if you specify it for an existing data set.

The use of a storage class can be protected by RACF.

References: See *MVS/XA Interactive Storage Management Facility User's Guide* for information on how to use ISMF to view your installation-defined storage classes.

Syntax:

```
STORCLAS=storage-class-name
```

Subparameter Definition

storage-class-name
Specifies the name of a storage class to be used for storage of the data set.

The name, one to eight characters, is defined by the storage administrator at your installation.

Defaults

If you do not specify STORCLAS for a new data set and the storage administrator has provided an installation-written automatic class selection (ACS) routine, the ACS routine may select a storage class for the data set. Check with your storage administrator to determine if an ACS routine will select a storage class for the new data set, in which case you do not need to specify STORCLAS.

Overrides

No attributes in the storage class can be overridden by JCL parameters.

An ACS routine can override the storage class that you specify on the STORCLAS parameter.

Relationship to Other Parameters

If the storage administrator has specified GUARANTEED_SPACE=YES in the storage class, then volume serial numbers you specify on the VOLUME=SER parameter override the volume serial numbers used by SMS. Otherwise, volume serial numbers are ignored.

Do not code the following DD parameters with the STORCLAS parameter.

*	DYNAM	UNIT=AFF
DATA	QNAME	VOLUME=REF
DDNAME		

Examples of the STORCLAS Parameter

```
//SMSDS1   DD  DSNAME=MYDS1.PGM,STORCLAS=SCLAS01,DISP=(NEW,KEEP)
```

In the example, SMS uses the attributes in the storage class named SCLAS01 for the storage service level of the data set. Note that installation-written ACS routines may select a management class and data class and can override the specified storage class.

```
//SMSDS2   DD  DSNAME=MYDS2.PGM,STORCLAS=SCLAS02,DISP=(NEW,KEEP),
//             VOLUME=SER=(223344,224444)
```

In the example, SMS uses the attributes in the storage class named SCLAS02 for the storage service level of the data set. Also, if the storage administrator has specified GUARANTEED_SPACE=YES in the storage class, VOLUME=SER can be coded and the data set will reside on the specified volumes. (However, if space is not available on the volumes, the job step fails.) Note that installation-written ACS routines may select a management class and data class and can override the specified storage class.

SUBSYS Parameter

Parameter Type: Keyword, optional

Purpose: Use the SUBSYS parameter to request a subsystem to process this data set and, optionally, to specify parameters defined by the subsystem.

Do not use the SUBSYS parameter for an SMS-managed data set (one with an assigned storage class).

In a loosely-coupled multiprocessing environment, the requested subsystem must be defined on all processors that could interpret this DD statement.

References: For more information on the SUBSYS parameter and subsystem-defined parameters, refer to the documentation for the requested subsystem.

Syntax:

```
SUBSYS={subsystem-name                                    }
       {(subsystem-name[,subsystem-subparameter]...)      }
```

Single Subparameter: You can omit the parentheses if you code only the subsystem-name.

Number of Subparameters: Code up to 254 subsystem-subparameters, if needed.

Multiple Subparameters: When the parameter contains more than the subsystem-name, separate the subparameters by commas and enclose the subparameter list in parentheses. For example, SUBSYS=(XYZ,1724,DT25).

Positional Subparameters: If you omit a subparameter that the subsystem considers positional, code a comma in its place.

Special Characters: When a subparameter contains special characters, enclose the subparameter in apostrophes. For example, SUBSYS=(XYZ,1724,'KEY=Y').

Code each apostrophe that is part of a subparameter as two consecutive apostrophes. For example, code O'Day as SUBSYS=(XYX,1724,'NAME=O''DAY').

Continuation onto Another Statement: Enclose the subparameter list in only one set of parentheses. End each statement with a comma after a complete subparameter. For example:

```
//DS1  DD  DSNAME=DATA1,SUBSYS=(XYZ,1724,'KEY=Y',
//           DT25,'NAME=O''DAY')
```

Subparameter Definition

subsystem-name
Identifies the subsystem. The subsystem name is 1 through 4 alphanumeric or national ($, #, @) characters; the first character must be alphabetic or national ($, #, @). The subsystem must be available in the installation.

subsystem-subparameter
Specifies information needed by the subsystem. A subparameter consists of alphanumeric, national ($, #, @), or special characters.

Relationship to Other Parameters

Do not code the following DD parameters with the SUBSYS parameter.

*	DYNAM
AMP	FLASH
BURST	MODIFY
CHARS	QNAME
COPIES	SPACE
DATA	SYSOUT
DDNAME	

The specified subsystem can define other parameters that must not be coded with the SUBSYS parameter.

Ignored but Permitted DD Parameters: If the following DD parameters are specified, they are checked for syntax and ignored:

FCB
UNIT

DISP Parameter: The system checks the DISP status subparameter for syntax, but always indicates a status of MOD to the subsystem. If the DISP normal or abnormal termination subparameter is CATLG or UNCATLG, the system allocates the appropriate catalog to the subsystem.

DUMMY Parameter: If DUMMY is specified with SUBSYS, the subsystem checks the syntax of the subsystem subparameters. If they are acceptable, the system treats the data set as a dummy data set.

When This Statement Overrides a Procedure Statement: If SUBSYS appears on a DD statement that overrides a DD statement in a cataloged or in-stream procedure, the following occurs:

- The system ignores a UNIT parameter, if specified, on the overridden DD statement.
- The system nullifies a DUMMY parameter, if specified, on the overridden DD statement.

DD: SUBSYS

Examples of the SUBSYS Parameter

```
//DD1  DD  DSNAME=ANYDS,DISP=OLD,SUBSYS=ABC
```

The DD statement asks subsystem ABC to process data set ANYDS.

```
//DD1 DD  DSNAME=ANYDS,DISP=OLD,SUBSYS=(XYZ2,
//         'KEYWORD=DATA VALUE1')
```

The DD statement asks subsystem XYZ2 to process data set ANYDS. The system passes the subparameter KEYWORD=DATA VALUE1 to the subsystem. The parameter is enclosed in apostrophes because it contains an equal sign and a blank, which are special characters.

```
//DD1 DD DSNAME=ANYDS,DISP=OLD,SUBSYS=(XYZ2,IKJ2,
//        'NAME=''MODULE1''','DATE=4/11/86')
```

The DD statement asks subsystem XYZ2 to process the data set ANYDS. The system passes three subparameters to the subsystem: IKJ2, NAME='MODULE1' and DATE=4/11/86. Note that the character string MODULE1 is passed to the subsystem enclosed in apostrophes.

SYSOUT Parameter

Parameter Type: Keyword, optional

Do not use the SYSOUT parameter for an SMS-managed data set (one with an assigned storage class).

Purpose: Use the SYSOUT parameter to identify this data set as a system output data set, usually called a sysout data set. The SYSOUT parameter also:

- Assigns this sysout data set to an output class. The attributes of each output class are defined during JES initialization; the attributes include the device or devices for the output class.
- Optionally requests an external writer to process the sysout data set rather than JES. An external writer is an IBM- or installation-written program.
- Optionally identifies the forms on which the data set is to be printed or punched.
- Optionally refers to a JES2 /*OUTPUT statement for processing parameters.

The sysout data set is processed according to the following processing options, in override order:

- The options specified on this sysout DD statement.
- The options specified on a referenced OUTPUT JCL statement.
- The options specified on a referenced JES2 /*OUTPUT statement or on a JES3 //*FORMAT statement.
- The installation default options for the requested output class.

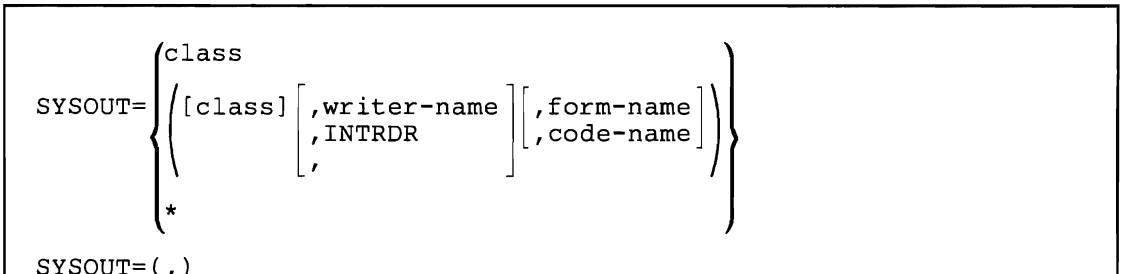
Note: If a sysout data set has the same class as the JOB statement MSGCLASS parameter, the job log appears on the same output listing as this sysout data set.

Output Classes: The installation should maintain a list of available output classes and their attributes. Some classes should be used for most printing and punching, but others should be reserved for special processing. Each class is processed by an output writer. The system operator starts the output writers for the commonly used output classes. If you plan to specify a special output class, ask the operator to start the output writer for that class. If the writer is not started before the job produces the sysout data set, the data set is retained until the writer is started.

References: For information on output writers and external writers, see *SPL: System Modifications.*

DD: SYSOUT

Syntax:



- You can omit the parentheses if you code only a class.
- All of the subparameters are positional. Code a comma to indicate an omitted subparameter as follows:
  - If you omit the class, code a comma to indicate the omission. For example, when other subparameters follow, code SYSOUT=(,INTRDR,FM26). When other subparameters do not follow, code a null class as SYSOUT=(,).
  - If you omit a writer-name but code a form-name or code-name, code a comma to indicate the omission. For example, SYSOUT=(A,,FM26).
  - Omission of the third subparameter does not require a comma. For example, SYSOUT=A or SYSOUT=(A,INTRDR).

Subparameter Definition

class
Identifies the output class for the data set. class is one character: A through Z or 0 through 9. The attributes of each output class are defined during JES initialization; specify the class with the desired attributes.

*****
Requests the output class in the MSGCLASS parameter on the JOB statement.

In a JES2 system, the dollar-sign ($) can also be used to request the output class in the MSGCLASS parameter on the JOB statement. The $ is provided only for compatibility with previous releases of JES2. The asterisk (*) value should be used instead of the $.

(,)
Specifies a null class. A null class must be coded to use the CLASS parameter on a referenced OUTPUT JCL statement.

writer-name
Identifies the member name (1 to 8 alphanumeric characters) of an installation-written program in the system library that the external writer loads to write the output data set.

Do not code STDWTR as a writer-name. STDWTR is reserved for JES and used as a parameter in the MVS operator's MODIFY command.

In a JES3 system, do not code NJERDR as a writer-name. NJERDR is reserved for JES3.

INTRDR
Tells JES that this sysout data set is to be sent to the internal reader as an input job stream.

form-name
Identifies the print or punch forms. form-name is 1 through 4 alphanumeric or national ($, #, @) characters.

code-name
Identifies an earlier JES2 /*OUTPUT statement from which JES2 is to obtain processing characteristics. The code-name must be the same as the **code** parameter on the JES2 /*OUTPUT statement.

Note:

- code-name is supported only on JES2 systems.
- Do not specify the code-name subparameter when the job or job step contains a default OUTPUT JCL statement.

Defaults

In a JES2 system, if you do not specify a class on this DD statement or a referenced OUTPUT JCL statement, JES2 assigns the sysout data set to output class A.

If you do not code a writer-name subparameter on this DD statement or a referenced OUTPUT JCL statement, the installation's job entry subsystem processes the sysout data set.

If you do not code a form-name subparameter on this DD statement or a referenced OUTPUT JCL statement, JES uses an installation default specified at initialization.

Overrides

The class subparameter of the DD statement SYSOUT parameter overrides an OUTPUT JCL CLASS parameter. On the DD statement, you must code a null class in order to use the OUTPUT JCL CLASS parameter; for example:

```
//OUTDS DD SYSOUT=(,),OUTPUT=*.OUT1
```

The writer-name subparameter of the DD statement SYSOUT parameter overrides an OUTPUT JCL WRITER parameter.

The form-name subparameter of the DD statement SYSOUT parameter overrides an OUTPUT JCL FORMS parameter. Note that the SYSOUT form-name subparameter can be only four characters maximum while both the OUTPUT JCL FORMS form-name and the JES initialization default form names can be eight characters maximum.

DD: SYSOUT

Relationship to Other Parameters

Do not code the following DD parameters with the SYSOUT parameter.

*	DSNAME	PROTECT
AMP	DYNAM	QNAME
CHKPT	EXPDT	RETPD
DATA	LABEL	SUBSYS
DDNAME	LIKE	VOLUME
DISP	MSVGP	

Ignored Parameters: Because JES allocates sysout data sets, the UNIT and SPACE parameters are ignored, if coded on a sysout DD statement.

Parameters on Procedure DD Statements that are Overridden: When an overriding DD statement contains a SYSOUT parameter, the system ignores a UNIT parameter on the overridden DD statement in the cataloged or in-stream procedure.

If the overridden DD statement contains a DSNAME parameter, the system issues a JCL warning message.

SYSOUT and DEST Subparameters: Do not code the SYSOUT writer-name subparameter when coding a DEST userid subparameter. These subparameters are mutually exclusive. You can code:

```
//VALID1  DD  SYSOUT=D,DEST=(node,userid)
//VALID2  DD  SYSOUT=(D,writer-name),DEST=(node)
```

With DCB Subparameters: JES2 ignores DCB=PRTSP=2 on a DD statement that also contains a SYSOUT parameter.

For JES2, it is not necessary to select a specific BLKSIZE on the DCB parameter for performance reasons because the subsystem selects its own blocking and overrides the DCB BLKSIZE specification.

INTRDR with OUTPUT Parameter: Do not code an OUTPUT parameter when the writer-name subparameter is INTRDR.

Relationship to Other Control Statements

A sysout DD statement can directly or indirectly reference an OUTPUT JCL statement. The parameters on the referenced OUTPUT JCL statement combine with the parameters on the sysout DD statement to control the processing of the sysout data set. See "OUTPUT Parameter" on page 10-128 and Chapter 17, "OUTPUT JCL Statement."

SYSOUT cannot specify a code-name subparameter in a job or job step that contains an OUTPUT JCL statement; in this case, JES2 treats the third subparameter as a form-name, instead of a reference to a JES2 /*OUTPUT statement.

Backward References: Do not refer to a earlier DD statement that contains a SYSOUT parameter.

Starting an External Writer when Requested

When a statement supplying processing options for a sysout data set specifies an external writer, the writer must be started before it can print or punch the data set. The writer is started by a system command from the operator or in the input stream. If the writer is not started before the job produces the sysout data set, the data set is retained until the writer is started.

Held Classes in a JES2 System

An installation option at JES2 initialization determines if both the class for the sysout data set and the class for the job's messages must be held in order for a sysout data set to be held.

A sysout data set is held in the following cases:

- The sysout DD statement contains HOLD=YES.
- The sysout DD statement does not contain a HOLD parameter or contains HOLD=NO but requests a class that the installation defined as held and defined as:
  - Not requiring the message class to be a held class in order for the sysout data set to be held. The JOB statement MSGCLASS parameter can specify any class.
  - Requiring the message class to be a held class in order for the sysout data set to be held. The JOB MSGCLASS parameter must also specify a held class.

A sysout data set is not held in the following cases:

- The sysout DD statement does not contain a HOLD parameter or contains HOLD=NO and requests:
  - A class that the installation defined as not held.
  - A class that the installation defined as held and defined as requiring the message class to be a held class in order for the sysout data set to be held. The JOB MSGCLASS parameter must specify a class that is not held.

Contact the installation to find out if holding the sysout class depends on a held MSGCLASS class.

Held Classes in a JES3 System

If CLASS specifies a class-name that is defined to JES3 as a held class for the output service hold queue (Q=HOLD), all of the new output characteristics might not be included in the data set on the writer queue when (1) the data set is moved from the hold queue to the output service writer queue (Q=WTR), (2) the data set includes an OUTPUT JCL statement, and (3) the NQ= or NCL= keyword is used.

For more information, see *MVS/XA SPL: JES3 Initialization and Tuning*.

DD: SYSOUT

Significance of Output Classes

To print this sysout data set and the messages from your job on the same output listing, code one of the following:

- The same output class in the DD SYSOUT parameter as in the JOB MSGCLASS parameter.
- DD SYSOUT=* to default to the JOB MSGCLASS output class.
- DD SYSOUT=(,) to default to one of the following:
  1. The CLASS parameter in an explicitly or implicitly referenced OUTPUT JCL statement. In this case, the OUTPUT JCL CLASS parameter should specify the same output class as the JOB MSGCLASS parameter.
  2. The JOB MSGCLASS output class, if no OUTPUT JCL statement is referenced or if the referenced OUTPUT JCL statement contains CLASS=*.

Examples of the SYSOUT Parameter

```
//DD1     DD      SYSOUT=P
```

In this example, the DD statement specifies that JES is to write the sysout data set to the device handling class P output.

```
//JOB50   JOB     ,'C. BROWN',MSGCLASS=C
//STEP1   EXEC    PGM=SET
//DDX     DD      SYSOUT=C
```

In this example, DD statement DDX specifies that JES is to write the sysout data set to the device handling class C output. Because the SYSOUT parameter and the MSGCLASS parameter specify the same class, the messages from this job and the sysout data set can be written to the same device.

```
//STEP1  EXEC   PGM=ANS
//OT1    OUTPUT DEST=NYC
//OT2    OUTPUT DEST=LAX
//OT3    OUTPUT COPIES=5
//DSA    DD     SYSOUT=H,OUTPUT=(*.OT2,*.OT1,*.OT3)
```

In this example, the DD statement combines with the three referenced OUTPUT JCL statements to create three separate sets of output:

1. DSA combines with OT1 to send the sysout data set to NYC.
2. DSA combines with OT2 to send the sysout data set to LAX.
3. DSA combines with OT3 to print five copies of the data set locally on the printer used for output class H.

Note that the output references can be in any order.

```
//DD5    DD     SYSOUT=(F,,2PRT)
```

In this example, the DD statement specifies that JES is to write the sysout data set to the device handling class F output. The data set is to be printed or punched on forms named 2PRT.

```
//JOB51  JOB    ACCT123,MAEBIRD,MSGCLASS=B
//STEP1  EXEC   PGM=RPTWTR
//OUT1   OUTPUT CLASS=*
//REPORT DD     SYSOUT=(,),OUTPUT=*.OUT1
```

In this example, JES processes the sysout data set defined in DD statement REPORT in output class B. Because SYSOUT specifies a null class, the CLASS parameter in the explicitly referenced OUTPUT JCL statement is used. That CLASS parameter specifies the MSGCLASS output class.

TERM Parameter

Parameter Type: Keyword, optional

Do not use the TERM parameter for an SMS-managed data set (one with an assigned storage class).

Purpose: Use the TERM parameter to indicate to the system that a data set is coming from or going to a terminal for a TSO user.

Syntax:

```
TERM=TS
```

Subparameter Definition

TS

In a **foreground job** submitted by a TSO user, indicates that the input or output data set is coming from or going to a TSO userid.

In a **background or batch job**, the system either:

- Ignores the TERM=TS parameter, when it appears with other parameters.
- Fails the TERM=TS parameter with an allocation error, when it appears by itself. (The system bypasses this error if SYSOUT=* is coded with TERM=TS.)

Relationship to Other Parameters

Do not code the following DD parameters with the TERM parameter.

*	DYNAM
AMP	PROTECT
DATA	QNAME
DDNAME	

Code only the DCB and SYSOUT parameters with the TERM parameter. The system ignores any other DD parameters.

Location in the JCL

In a foreground TSO job, a DD statement containing TERM=TS and a SYSOUT parameter begins an in-stream data set.

When concatenating DD statements, the DD statement that contains TERM=TS must be the last DD statement in a job step.

Examples of the TERM Parameter

```
//DD1   DD   TERM=TS
```

In a foreground job submitted from a TSO userid, this DD statement defines a data set coming from or going to the TSO userid.

```
//DD1   DD   TERM=TS,SYSOUT=*
```

In a background or batch job, the system ignores TERM=TS and recognizes a sysout data set. (An allocation error occurs if SYSOUT=* is not coded with TERM=TS.)

```
//DD3   DD   UNIT=3400-5,DISP=(MOD,PASS),TERM=TS,LABEL=(,NL),
//           DCB=(LRECL=80,BLKSIZE=80)
```

In a foreground job, the system ignores all of the parameters in this example except TERM and DCB. In a batch job, the system ignores only the TERM parameter.

UCS Parameter

Parameter Type: Keyword, optional

Purpose: Use the UCS parameter to identify:

- The universal character set (UCS) image JES is to use in printing this sysout data set.
- A print train (print chain or print band) JES is to use in printing this sysout data set on an impact printer.
- A character-arrangement table for this sysout data set printed on 3800 Printing Subsystem in a JES2 system. In this use, the UCS parameter acts like a CHARS parameter.

The UCS image specifies the special character set to be used. JES loads the image into the printer's buffer. The UCS image is stored in SYS1.IMAGELIB. IBM provides the special character set codes in Figure 10-2.

References: For more information on the UCS parameter, see *System-Data Administration.*

Syntax:

```
UCS={character-set-code                                    }
    {(character-set-code [,FOLD] [,VERIFY])                }
                         [,   ]
```

- You can omit the parentheses if you code only a character-set-code.
- All of the subparameters are positional. If you omit FOLD but code VERIFY, code a comma to indicate the omission. For example, UCS=(AN,,VERIFY).

Subparameter Definition

character-set-code
Identifies a universal character set. The character-set-code is 1 through 4 alphanumeric or national ($, #, @) characters. See Figure 10-2 for IBM standard special character set codes.

FOLD
Requests that the chain or train for the universal character set be loaded in fold mode. Fold mode is described in *IBM 2821 Control Unit Component Description.* Fold mode is most often used when upper- and lower-case data is to be printed only in uppercase.

Note: JES2 and JES3 do not support the FOLD subparameter. For JES2, the FOLD option is specified in the UCS image for JES2-controlled printers. See *MVS/XA System-Data Administration.*

VERIFY

Requests that, before the data set is printed, the operator verify visually that the character set image is for the correct chain or train. The character set image is displayed on the printer before the data set is printed.

1403	3203 Model 5	3211	Characteristics
AN	AN	A11	Arrangement A, standard EBCDIC character set, 48 characters
HN	HN	H11	Arrangement H, EBCDIC character set for FORTRAN and COBOL, 48 characters
		G11	ASCII character set
PCAN	PCAN		Preferred alphanumeric character set, arrangement A
PCHN	PCHN		Preferred alphanumeric character set, arrangement H
PN	PN	P11	PL/I alphanumeric character set
QN	QN		PL/I preferred alphanumeric character set for scientific applications
QNC	QNC		PL/1 preferred alphanumeric character set for commercial applications
RN	RN		Preferred character set for commercial applications of FORTRAN and COBOL
SN	SN		Preferred character set for text printing
TN	TN	T11	Character set for text printing, 120 characters
XN			High-speed alphanumeric character set for 1403, Model 2
YN			High-speed preferred alphanumeric character set for 1403, Model N1

Note: Where three values exist (for the 1403, 3211, and 3203 Model 5 printers), code any one of them. JES selects the set corresponding to the device on which the data set is printed.

Not all of these character sets may be available at your installation. Also, an installation can design character sets to meet special needs and assign a unique code to them. Follow installation procedures for using character sets.

Figure 10-2. Special Character Sets for the 1403, 3203 Model 5, and 3211 Printers

Defaults

If you do not code the UCS parameter, the system checks the UCS image in the printer's buffer; if it is a default image, as indicated by its first byte, JES uses it. If it is not a default image, JES loads the UCS image that is the installation default specified at JES initialization.

On an impact printer, if the chain or train does not contain a valid character set, JES asks the operator to specify a character set and to mount the corresponding chain or train.

Overrides

For printing on a printer with the UCS feature, the UCS parameter on a sysout DD statement overrides an OUTPUT JCL UCS parameter. For printing on a 3800, a CHARS parameter on the sysout DD statement or the OUTPUT JCL statement overrides all UCS parameters.

For a data set scheduled to the Print Services Facility (PSF), the PSF uses the following parameters, in override order, to select the font list:

1. Font list in the library member specified by an OUTPUT JCL PAGEDEF parameter.
2. DD CHARS parameter.
3. OUTPUT JCL CHARS parameter.
4. DD UCS parameter.
5. OUTPUT JCL UCS parameter.

6. JES installation default for the device.
7. Font list on the PAGEDEF parameter in the PSF cataloged procedure.

See "PAGEDEF Parameter" on page 17-51 for more information.

Relationship to Other Parameters

Do not code the following DD parameters with the UCS parameter.

*	DYNAM
AMP	KEYOFF
DATA	PROTECT
DDNAME	QNAME

Do not code the UCS parameter with the DCB subparameters CYLOFL, INTVL, RESERVE, and RKP.

Do not code the FOLD and VERIFY subparameters on the same statement with a SYSOUT parameter; the system ignores FOLD and VERIFY for a sysout data set.

Using Special Character Sets

To use a special character set, SYS1.IMAGELIB must contain an image of the character set, and the chain or train for the character set must be available. IBM provides standard special character sets, and the installation may provide user-designed special character sets.

Examples of the UCS Parameter

```
//DD1 DD UNIT=1403,UCS=(YN,,VERIFY)
```

In this example, the DD statement requests a 1403 Printer. The UCS parameter requests the chain or train for special character set code YN. Because VERIFY is coded, the system will display the character set image on the printer before the data set is printed.

```
//DD2  DD  SYSOUT=G,UCS=PN
```

In this example, the DD statement requests the device for output class G. If the device is a printer with the UCS feature, the system loads the UCS image for code PN. If the device is an impact printer, the system asks the operator to mount the chain or train for PN, if it is not already mounted. If the device is a 3800, the system uses the UCS subparameter to select the character-arrangement table. Otherwise, the system ignores the UCS parameter.

UNIT Parameter

Parameter Type: Keyword, optional

Note: With SMS, you do not need to use the UNIT parameter to specify a device for an SMS-managed data set. Use the STORCLAS parameter (described on page 10-156) or let an installation-written automatic class selection (ACS) routine select a storage class for the data set.

Also with SMS, for a non-SMS-managed data set, if your storage administrator has set a system default unit under SMS, you do not need to specify UNIT. Check with your storage administrator.

Purpose: Use the UNIT parameter to ask the system to place the data set on:

- A specific device.
- A certain type or group of devices.
- The same device as another data set.

The UNIT parameter can also tell the system how many devices to assign and request that the operator should defer mounting the volume until the data set is opened.

Syntax:

```
{UNIT=(device-number  [,unit-count  ]  [,DEFER])}
      (device-type    [,P           ]
      (group-name     [,            ]
{UNIT=AFF=ddname                                }
```

- You can omit the parentheses if you code only the first subparameter.
- All of the subparameters are positional. If you omit unit-count or P but code DEFER, code a comma to indicate the omission; one device is assigned to the data set. For example, UNIT=(3420,,DEFER).

Subparameter Definition

device-number

Identifies a particular device. device-number is the 3-character hexadecimal number for the device.

Warning: Do not identify a device by its number unless absolutely necessary. Specifying a device number limits device assignment; it will delay a job if another job is using the device.

In a JES3 system, if any DD UNIT parameter in a job specifies a device-number for a device that is JES3-managed or jointly JES3/MVS managed, the JES3 //*MAIN statement must contain a SYSTEM parameter.

However, for a permanently mounted direct access device, such as a 3350 Direct Access Storage, specifying UNIT=3350 and a volume serial number in the VOLUME=SER parameter has the same result as specifying a device number in the UNIT parameter.

device-type
Requests a device by its generic name, which is an IBM-supplied name that identifies a device by its machine type and model. For example, UNIT=3350.

When a device-type name contains a hyphen, do not enclose it in apostrophes, for example, UNIT=3400-5.

Obtain a list of device types from your installation.

Rules for Certain Devices:

- For a 3330 Disk Storage Model 11, code UNIT=3330-1.
- If your installation has 3340 Direct Access Storage Facilities both with and without the Fixed Head feature, do not code the device type in the UNIT parameter. Instead, code the device number or a group-name.
- For a 3480 Magnetic Tape Subsystem in compatibility mode, code UNIT=3400-9 or a group-name.
- For mass storage volumes, code UNIT=3330V.

group-name
Requests a group of devices by a symbolic name. The installation must have assigned the name to the device(s) during system generation or IBM must have assigned the name. The group-name is 1 through 8 alphanumeric characters.

Group Names: A group-name can identify a single device or a group of devices. A group can consist of devices of the same or different types. For example, a group can contain both direct access and tape devices.

Allocation from Groups: The system assigns any available device from the group. If a group consists of only one device, the system assigns that device. If the group consists of more than one device type, the units requested are allocated from the same device type. For example, if GPDA contains 3330 Disk Storage and 3350 Direct Access Storage devices, a request for two units would be allocated to two 3330s or to two 3350s.

Extending Data Set: If a data set that was created using the group-name subparameter is to be extended, the system allocates additional devices of the same type as the original devices. However, the additional devices may not necessarily be from the same group.

SYSALLDA: IBM assigned group-names include SYSALLDA, which contains all direct access devices defined to the system.

unit-count
Specifies the number of devices for the data set. The unit-count is a decimal number from 1 through 59.

Number of Devices Allocated: The system uses the unit-count to determine how many devices to allocate. However, if you also specify the VOLUME parameter, the system

uses the **greatest** of the following numbers to determine how many devices and volumes to allocate:

- unit-count specified in the UNIT parameter
- volume-count specified in the VOLUME parameter
- number of serial numbers implicitly or explicitly specified

You may also receive more devices than the unit-count requests if you specify VOLUME=REF or a permanently resident or reserved volume. Also, if two DD statements in a step request the same volume and either DD statement requests any other volume(s), the system assigns an additional device.

Unit Count for Received or VOLUME=REF Data Sets: The system assigns one device when the DD statement receives a passed data set or refers in a VOLUME=REF subparameter to a cataloged data set or earlier DD statement for volume and unit information. Code a unit-count subparameter if the data set needs more than one device.

Unit Count for Mass Storage Volumes: For mass storage volumes, specify a unit-count that is less than the volume-count in order to extend a multivolume data set to a non-mounted volume. If an old multivolume data set resides on a group of mass storage volumes, specify a unit-count equal to the number of volumes for the data set or specify P for parallel mount; otherwise, in a JES3 environment, the last volume containing the data set is left mounted exclusive to the operating system.

Unit Count when Device Number Specified: When the first subparameter requests a specific device, the unit count must be 1 or omitted. A unit count greater than 1 is an error.

P

Asks the system to allocate the same number of devices as requested in the VOLUME volume-count or SER subparameter, whichever is higher. Thus, all volumes for the data set are mounted in parallel.

DEFER

Asks the system to assign the data set to device(s) but requests that the volume(s) not be mounted until the data set is opened. To defer mounting, DEFER must be specified or implied for all DD statements that reference the volume.

DEFER when Data Set is Never Opened: If you request deferred mounting of a volume and the data set on that volume is never opened by the processing program, the volume is never mounted during the job step.

Restrictions on DEFER: Do not code DEFER:

- For a new data set on direct access. The system ignores DEFER.
- On a SYSCKEOV DD statement.
- For volumes of a mass storage volume group (MSVGP) when the step contains new data set requests for the same mass storage volume group. The delay can cause volume conflicts within the job or between jobs and so cause poor performance.

AFF = ddname

Requests that the system allocate different data sets residing on different, removable volumes to the same device during execution of the step. This request is called **unit affinity**. The ddname is the ddname of an earlier DD statement in the same step.

Use unit affinity to reduce the number of devices used in a job step: request that an existing data set be assigned to the same device(s) as another existing data set.

Restrictions on UNIT = AFF: Do not code UNIT = AFF with the following:

- DISP = NEW if the data set referenced in the AFF subparameter is on direct access.
- DD * or DD DATA.
- FREE = CLOSE in the DD statement referenced in the AFF subparameter.

Overrides

If you code SYSOUT and UNIT on the same statement, the SYSOUT parameter overrides the UNIT parameter.

The system also obtains device information when the system obtains volume serial information from:

- A VOLUME = REF = dsname reference to an earlier data set.
- A VOLUME = REF = ddname reference to an earlier DD statement.
- The volume(s) for a passed data set.
- The catalog for a cataloged data set.

However, you can override the retrieved device information if the device you specify is a subset of the retrieved device information. For example, if the retrieved unit grouping is 3350, and the specified unit subparameter is 3350A (a subset of 3350), then the system allocates from the devices contained in 3350A.

Relationship to Other Parameters

Do not code the follcwing DD parameters with the UNIT parameter.

*	DYNAM
DATA	QNAME
DDNAME	

Do not code the UNIT DEFER subparameter on a SYSCKEOV DD statement.

UNIT = AFF and Other Parameters: Do not code DISP = NEW with UNIT = AFF = ddname if the referenced data set for DD statement ddname resides on a direct access device. If coded, the system terminates the job. If the referenced data set can be allocated to either tape or direct access devices, the system allocates both requests to tape devices.

Do not code UNIT = AFF with the STORCLAS parameter.

Do not code UNIT = AFF on a DD * or DD DATA statement or on a DD statement containing a SUBSYS parameter. The system ignores UNIT = AFF and defaults the device to SYSALLDA.

Do not code UNIT = AFF = ddname when DD statement ddname contains FREE = CLOSE.

Location in the JCL

When a DD statement contains a UNIT=AFF=ddname parameter, the DD statement referenced in the AFF subparameter must be earlier in the job step; otherwise, the system treats the DD statement containing UNIT=AFF as a DD DUMMY statement. For example, code:

```
//STEP  EXEC  PGM=TKM
//DD1   DD    DDNAME=DD5
//DD2   DD    DSNAME=A,DISP=OLD
//DD3   DD    DSNAME=C,DISP=SHR,UNIT=AFF=DD1
//DD5   DD    DSNAME=B,DISP=SHR
```

Examples of the UNIT Parameter

```
//STEP2  EXEC PGM=POINT
//DDX  DD  DSNAME=EST,DISP=MOD,VOLUME=SER=(42569,42570),
//     UNIT=(3330,2)
//DDY  DD  DSNAME=ERAS,DISP=OLD,UNIT=3330-1
//DDZ  DD  DSNAME=RECK,DISP=OLD,
//     VOLUME=SER=(40653,13262),UNIT=AFF=DDX
```

DD statement DDX requests two 3330 Disk Storage devices, DD statement DDZ requests the same two devices as DDX. Note that the operator will have to change volumes on the two 3330 devices during execution of the job step.

DD statement DDY requests one 3330 Disk Storage Model 11.

```
//DD1  DD  DSNAME=AAG3,DISP=(,KEEP),
//       VOLUME=SER=13230,UNIT=3400-5
```

This DD statement defines a new data set and requests that the system assign any 3420 Magnetic Tape Unit that can operate in 6250 BPI NRZI nine-track format.

```
//DD2  DD  DSNAME=X.Y.Z,DISP=OLD,UNIT=(,2)
```

This DD statement defines a cataloged data set and requests that the system assign two devices to the data set. The system obtains the device type from the catalog.

```
//DD3  DD  DSNAME=COLLECT,DISP=OLD,
//       VOLUME=SER=1095,UNIT=(3330,,DEFER)
```

This DD statement defines an existing data set that resides on a direct access volume and requests that the system assign a 3330 Disk Storage. Because DEFER is coded, the volume will not be mounted until the data set is opened.

```
//STEPA  DD  DSNAME=FALL,DISP=OLD,UNIT=237
```

For this data set, the system retrieves the volume and device type from the catalog. The UNIT parameter, by specifying device 237, overrides the catalog device type; however, device 237 must be the same type as the device stated in the catalog.

VOLUME Parameter

Parameter Type: Keyword, optional

Note: With SMS, you do not need to use the VOLUME parameter to specify volumes for new data sets. See the DATACLAS parameter described on page 10-42 and the STORCLAS parameter described on page 10-156.

Also with SMS, if the storage administrator has specified a system default unit, then the system uses the volumes associated with the default unit. In this case, you do not need to code the VOLUME parameter. Check with your storage administrator.

Purpose: Use the VOLUME parameter to identify the volume or volumes on which a data set resides or will reside. You can request:

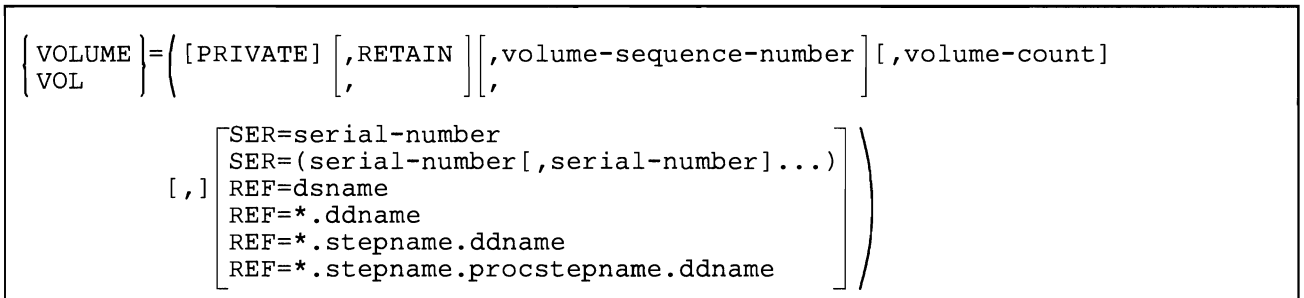
- A private volume
- Retention of the volume
- A specific volume by serial number
- The same volume that another data set uses

You can also specify which volume of a multivolume data set is to be processed first and, for an output data set, the number of volumes required.

A **nonspecific volume request** is a DD statement for a new data set that can be assigned to any volume or volumes. To make a nonspecific volume request for a new data set, either:

- Omit the VOLUME parameter.
- Code a VOLUME parameter but omit a SER or REF subparameter.

Syntax:



Single Subparameter: You can omit the parentheses if you code only PRIVATE or only a keyword subparameter. For example, VOLUME=PRIVATE or VOLUME=SER=222001 or VOLUME=REF=DS1.

Positional Subparameters: The first four subparameters are positional. The last subparameter, SER or REF, is a keyword subparameter and must follow all positional subparameters. Code a comma to indicate an omitted positional subparameter as follows:

- If you omit PRIVATE and code RETAIN, code a comma before RETAIN. For example, VOLUME=(,RETAIN,2,3,SER=(222001,222002,222003)).
- Code a comma when RETAIN is omitted and the volume sequence number and volume count subparameters follow. For example, VOLUME=(PRIVATE,,2,3,SER=(222001,222002,222003)), and if PRIVATE is also omitted, VOLUME=(,,2,3,SER=(222001,222002,222003)).
- Code a comma when the volume sequence number is omitted and the volume count subparameter follows. For example, VOLUME=(,RETAIN,,3,SER=(222001,222002,222003)), and VOLUME=(PRIVATE,,,3,SER=(222001,222002,222003)), and VOLUME=(,,,3,SER=(222001,222002,222003)).
- Code a comma when the volume count is omitted, at least one other subparameter precedes it, and a keyword subparameter follows. For example, VOLUME=(,RETAIN,2,,SER=(222001,222002,222003)), and VOLUME=(,,2,,SER=(222001,222002,222003)).

Single SER Subparameter: You can omit the parentheses in the SER subparameter if you code only one serial number. For example, VOLUME=SER=222001.

Special Characters When a serial number in the SER subparameter contains special characters, other than hyphens, enclose it in apostrophes. For example, VOLUME=SER=(222001,222-02,'222/03').

When the dsname in the REF subparameter contains special characters, other than the periods used in a qualified name, enclose it in apostrophes. For example, VOLUME=REF='DS/284'.

Code each apostrophe that is part of the serial number or data set name as two consecutive apostrophes. For example, VOLUME=SER='O''HARE' or VOLUME=REF='DS''371'.

Subparameter Definition

PRIVATE
Requests a private volume. Private means that:

- The system is not to allocate an output data set to the volume unless the volume is specifically requested, such as in a VOLUME=SER subparameter.
- If tape, the volume is to be demounted after the data set is closed, unless RETAIN is also coded or the DD DISP parameter specifies PASS.
- If a demountable direct access volume, the volume is to be demounted after the data set is closed.

RETAIN
For a private tape volume, RETAIN requests that this volume is not to be demounted or rewound after the data set is closed or at the end of the step. For a public tape volume, RETAIN requests that this volume is to be retained at the device if it is demounted during the job.

RETAIN Support: RETAIN is supported only for tape volumes managed by the basic control program and by JES2. If coded on a DD statement for a tape data set on a JES3-managed device, JES3 ignores the parameter when issuing KEEP/RETAIN messages and when performing deallocation at the end of the job. However, if RETAIN is coded for a tape data set on a JES3-managed device and the tape volume is to be shared with a later step, JES3 designates the volume as retained.

RETAIN has no effect on the handling of direct access volumes.

Deallocation Despite RETAIN: Coding RETAIN does not ensure that the operator will not unload the volume or that the system will not deallocate and demount it for another job. Either can occur when the device on which the volume is mounted is not allocated to the job step that specified RETAIN or, for unlabeled tapes, when the volume requires verification.

volume-sequence-number

Identifies the volume of an existing multivolume data set to be used to begin processing the data set. The volume sequence number is a decimal number from 1 through 255; the first volume is identified as 1. The volume sequence number must be less than or equal to the number of volumes on which the data set exists; otherwise, the job fails.

If the volume sequence number is not specified but DCB=dsname is coded on a DD statement, the volume sequence number is copied from the label for the cataloged data set.

For new data sets, the system ignores the volume sequence number.

volume-count

Specifies the maximum number of volumes that an **output** data set requires. The volume count is a decimal number from 1 through 255. The total volume count for all DD statements in one job step cannot exceed 4095.

Code a volume count when a new data set will reside on 6 or more volumes. If the volume count is omitted or if the specified count is 1 through 5, the maximum number allowed is 5; if the specified count is 6 through 20, the maximum number allowed is 20: if the specified count is greater than 20, the maximum number allowed is a multiple of 15 plus 5. The maximum volume count for a non-VSAM or non-SMS-managed data set is 255. The maximum volume count for a VSAM or SMS-managed data set is 59.

Volume Count and Serial Numbers: If the volume count is greater than the number of volume serials coded in the SER subparameter, the system assigns other volumes to the remaining devices. If the volume count is smaller than the number of volume serials, the system ignores the volume count.

If a data set may need more volumes than the number of volume serials coded, specify a volume count equal to the total number of volumes that might be used. Requesting more volumes in the volume count will make sure that the data set can be written on more volumes if it exceeds the requested volumes.

If you do not code a volume count and volume serial number(s), the system can extend an existing cataloged data set that resides on a removable volume up to 20 volumes.

Volume Count for Nonspecific Requests: If the request is for a nonspecific, public volume on a direct access device, the system ignores the volume count and allocates the number of volumes in the UNIT unit count subparameter.

If the request is for a nonspecific, private volume, the system treats it like a specific request if the volume count is more than one and allocates the number of volumes in the volume count.

For more information on determining the number of volumes per request, see *SPL: System Modifications*.

SER = serial-number
SER = (serial-number[,serial-number]...)

For an SMS-managed data set, code the SER subparameter only if the storage administrator has specified GUARANTEED_SPACE = YES in the storage class of the data set. The volume serial numbers you specify on the VOLUME = SER parameter override the volume serial numbers used by SMS. The volume serial numbers must be assigned to the same storage group. If GUARANTEED_SPACE = YES is not in effect, SMS ignores any volume serial numbers you specify for SMS-managed data sets.

Identifies by serial number the volume(s) on which the data set resides or will reside. A volume serial number is 1 through 6 alphanumeric, national ($, #, @), or special characters; enclose a serial number that contains special characters, other than hyphens, in apostrophes. If the number is shorter than 6 characters, it is padded with trailing blanks.

You can code a maximum of 255 volume serial numbers on a DD statement.

Do not specify duplicate volume serial numbers in a SER subparameter. Each volume must have a unique volume serial number, regardless of whether it is a tape or disk volume.

Do not code a volume serial number as SCRTCH, PRIVAT, or Lnnnnn (L with five numbers); these are used in messages to ask the operator to mount a volume. Do not code a volume serial number as MIGRAT, which is used by DFHSM for migrated data sets. When using some typewriter heads or printer chains, a volume serial number may be unrecognizable if you code certain special characters.

For a permanently mounted direct access device, such as a 3350 Direct Access Storage, specifying a volume serial number and UNIT = 3350 has the same result as specifying a device number in the UNIT parameter.

REF = dsname
REF = *.ddname
REF = *.stepname.ddname
REF = *.stepname.procstepname.ddname

Tells the system to obtain volume serial numbers from another data set or an earlier DD statement.

dsname

Names a cataloged or passed data set. The system assigns this data set to the same volumes containing the cataloged or passed data set.

The dsname can be an alias name or a catalog name. The dsname cannot be a generation data group (GDG) name or a GDG member.

When the dsname contains special characters, other than the periods used in a qualified name, enclose it in apostrophes.

When dsname names a passed data set, it must appear on a DD statement before the receiving DD statement. (After a passed data set is received, the passed data set information is no longer available.)

***.ddname**
Asks the system to obtain the volume serial numbers from earlier DD statement ddname in the same job step.

***.stepname.ddname**
Asks the system to obtain the volume serial numbers from DD statement, ddname, in an earlier step, stepname, in the same job.

***.stepname.procstepname.ddname**
Asks the system to obtain the volume serial numbers from a DD statement in a cataloged or in-stream procedure. Stepname is the name of the job step that calls the procedure, procstepname is the name of the procedure step that contains the DD statement, and ddname is the name of the DD statement.

Referenced Data Set Not Opened: When REF refers to a DD statement in a previous step and the data set was not opened, the system allocates a device that has the widest range of eligibility to meet both DD statement requests. Thus, the system might allocate a device for which the referring data set is not eligible. To prevent this problem for tape data sets, always code the DCB DEN subparameter on a DD statement that you plan to reference.

References to Multivolume Tape Data Sets: When REF refers to a data set residing on more than one tape volume, the system allocates only the last volume. If this job step extends the data set to more volumes and this data set was a specific request, this new volume information is not available to following DD statements.

References to Multivolume Direct Access Data Sets: When REF refers to a data set that resides on more than one direct access volume, the system allocates all of the volumes.

If a DD statement that is requesting a new data set has a unit count and volume count greater than one but specifies no volume serial numbers, one volume is allocated. If a second DD statement within the same step requests the same data set, the same volume is allocated to it. If this job step extends the data set to more volumes, this new volume information is not available to the second DD statement.

Two or more DD statements in the same step can request the same data set. However, if the data set is extended to additional volumes in that step, the additional volume information is not available to the second or succeeding DD statements within the step.

References to DD Statements with UNIT Group Names: When REF refers to a DD statement containing a UNIT group-name subparameter, the system allocates a device of the same type actually used for the referenced data set, but not necessarily a device in the referenced group-name.

References to VSAM Data Sets: When REF refers to a multivolume VSAM data set, the system allocates a device of the same type as the first device type used for the referenced VSAM data set.

References to SMS-Managed Data Sets: When REF refers to an SMS-managed data set, SMS manages the new data set using the same storage class as the referenced data set. In this case, do not code the STORCLAS parameter on the DD statement.

Do Not Refer to In-Stream Data Sets: Do not refer to a DD *, DD DATA, or DD SYSOUT statement. The system ignores the reference and defaults the device name to SYSALLDA, which is the group name for all direct access devices defined to the system.

References to DUMMY Data Sets: If ddname refers to a DD DUMMY statement, the data set for this DD statement is also assigned a dummy status.

Label Type Picked up from Referenced Statement: When REF is coded, the system also copies the LABEL label type subparameter from the referenced DD statement.

Overrides

The volume sequence number overrides a DISP=MOD parameter. Thus, instead of starting at the end of the data set on the last volume, according to the MOD subparameter, processing of the data set begins with the volume indicated by the volume sequence number.

Relationship to Other Parameters

Do not code the following parameters with the VOLUME parameter.

BURST	DDNAME	MODIFY
CHARS	DYNAM	QNAME
COPIES	FLASH	SYSOUT

Do not code VOLUME=REF with the STORCLAS parameter.

Other DD Parameter Picked up from Referenced Statement: When REF is coded, the system also copies the LABEL label type subparameter from the referenced DD statement.

With Mass Storage Volumes For New Data Sets:

- The SPACE parameter is required when VOLUME=SER is coded.
- To guarantee allocation of a nonspecific request to SYSGROUP, specify VOLUME=PRIVATE or MSVGP=SYSGROUP.

For 3540 Diskette Input/Output Units: The VOLUME=SER, DCB=BUFNO, and DSID parameters on a DD * or DD DATA statement are ignored except when they are detected by a diskette reader as a request for an associated data set. See *IBM 3540 Programmer's Reference*.

VOLUME Parameter in a JES3 System

When you do not code a volume serial number, code PRIVATE if you want JES3 to manage the allocation. Otherwise, MVS manages the allocation.

RETAIN is ignored in a JES3 system.

DD: VOLUME

VOLUME Parameter for Optical Readers

For optical readers, if no volume serial number is specified, the system assumes VOLUME=SER=OCRINP.

VOLUME Parameter for Nonspecific Volume Requests

A nonspecific volume request can appear on a DD statement for a new data set; the data set is assigned to any volume or volumes. The nonspecific request is made through a VOLUME parameter that does not contain a SER or REF subparameter. The parameter can contain the following subparameters:

```
VOLUME=(PRIVATE,RETAIN,,volume-count)
```

Examples of the VOLUME Parameter

```
//DD1  DD  DSNAME=DATA3,UNIT=3340,DISP=OLD,
//       VOLUME=(PRIVATE,SER=548863)
```

The DD statement requests an existing data set, which resides on the direct access volume, serial number 548863. Since PRIVATE is coded, the system will not assign to the volume another data set for which a nonspecific volume request is made and will demount the volume at the end of the job.

```
//DD2  DD  DSNAME=QUET,DISP=(MOD,KEEP),UNIT=(3400-5,2),
//       VOLUME=(,,,4,SER=(96341,96342))
```

The DD statement requests an existing data set, which resides on two volumes, serial numbers 96341 and 96342. The VOLUME volume count subparameter requests four volumes, if required. Thus, if more space is required, the system can assign a third and fourth volume.

```
//DD3  DD  DSNAME=QOUT,UNIT=3400-5
```

The DD statement defines a data set that is created and deleted in the job step. By omission of the VOLUME parameter, the statement makes a nonspecific volume request, thereby asking the system to assign a suitable volume to the data set.

```
//DD4  DD  DSNAME=NEWDASD,DISP=(,CATLG,DELETE),UNIT=3350,
//       VOLUME=SER=335006,SPACE=(CYL,(10,5))
```

This new data set is assigned to volume serial number 335006, which is a permanently mounted volume on a particular 3350 Direct Access Storage. You can obtain the same space on the same volume in another way: Instead of specifying the volume serial number and UNIT=3350, you can specify the device number of the particular 3350 device in the UNIT parameter.

```
//OUTDD  DD  DSNAME=TEST.TWO,DISP=(NEW,CATLG),
//           VOLUME=(,,,3,SER=(333001,333002,333003)),
//           SPACE=(TRK,(9,10)),UNIT=(3330,P)
//NEXT   DD  DSNAME=TEST.TWO,DISP=(OLD,DELETE)
```

DD statement OUTDD creates a multivolume data set and catalogs it. If the data set does not require three volumes, it will reside on fewer volumes. DD statement NEXT then deletes the data set.

If the data set resides on fewer volumes than the number of volumes on which it is cataloged, the following messages appear in the job log when the system deletes the data set:

```
IEF285I   TEST.TWO                                          DELETED
IEF285I   VOL SER NOS=333001,333003.
IEF283I   TEST.TWO                                          NOT DELETED
IEF283I   VOL SER NOS=333002 1.
IEF283I   TEST.TWO                                          UNCATALOGED
IEF283I   VOL SER NOS=333001,333002,333003.
```

If the data set resides on all specified volumes, the following messages appear in the job log when the system deletes the data set:

```
IEF285I   TEST.TWO                                          DELETED
IEF285I   VOL SER NOS=333001,333002,333003.
```

```
//SMSDS2  DD  DSNAME=MYDS2.PGM,STORCLAS=SCLAS02,DISP=(NEW,KEEP),
//            VOLUME=SER=(223344,224444)
```

For a new SMS-managed data set, SMS uses the attributes in the storage class named SCLAS02 for the storage service level of the data set. Also, if the storage administrator has specified GUARANTEED_SPACE=YES in the storage class, VOLUME=SER can be coded and the data set will reside on the specified volumes. (However, if space is not available on the volumes, the job step fails.) Installation-written automatic class selection (ACS) routines select the data class and management class.

Chapter 11. Special DD Statements

Use special DD statements to specify private catalogs, private libraries, and data sets for storage dumps and checkpoints.

These special statements are arranged alphabetically in the following pages.

Syntax:

```
//ddname  DD keyword-parameter[,keyword-parameter]...  [comments]
```

Special ddnames

The special data sets are identified by the following ddnames:

JOBCAT
JOBLIB
STEPCAT
STEPLIB
SYSABEND
SYSCHK
SYSCKEOV
SYSIN
SYSMDUMP
SYSUDUMP

Code these ddnames only when you want the special data sets.

JOBCAT DD Statement

Purpose: Use the JOBCAT DD statement to define a private VSAM or integrated catalog facility user catalog for the duration of a job. The system searches the private catalog for data sets before it searches the master catalog or a private catalog associated with the first qualifier of a data set's name.

Do not use the JOBCAT DD statement in a job that references an SMS-managed data set. SMS only accesses SMS-managed data sets that are cataloged in a system catalog.

You cannot specify OS CVOLs as JOBCAT. Access to an OS CVOL is possible only with a special CVOL pointer in the master catalog.

References: For more information on VSAM data sets, see the *VSAM Administration Guide*.

Syntax:

```
//JOBCAT DD  DISP={OLD|SHR},DSNAME=private-catalog-name[,parameter]... [comments]
```

Parameters on JOBCAT DD Statements

Do not specify any unit or volume information. The system obtains the location of the private catalog from the master catalog. Do not specify FREE=CLOSE; CLOSE is ignored.

Relationship to STEPCAT DD Statement

A JOBCAT DD statement applies to any job step for which you do not specify a STEPCAT DD statement.

Relationship to Other Control Statements

Concatenating Job Catalogs: To specify more than one private catalog for a job:

- Code a JOBCAT DD statement.
- Immediately follow this statement with DD statements that define other private catalogs. Omit a ddname from these subsequent DD statements.

Location in the JCL

- Place the JOBCAT DD statement *after* the JOB statement and *before* the first EXEC statement.
- Place a JOBLIB DD statement, if coded, before a JOBCAT DD statement.

Example of the JOBCAT DD Statement

```
//EXAMPLE  JOB  WILLIAMS,MSGLEVEL=1
//JOBLIB   DD   DSNAME=USER.LIB,DISP=SHR
//JOBCAT   DD   DSNAME=LYLE,DISP=SHR
//         EXEC PGM=SCAN
```

In this example, the JOBCAT DD statement specifies a private catalog. The JOBCAT DD statement follows the JOBLIB DD statement.

JOBLIB DD Statement

Purpose: Use the JOBLIB DD statement to:

- Create a private library.
- Identify a private library that the system is to search for the program named in each EXEC statement PGM parameter in the job. If the system does not find the program in the private library, only then does the system search the system libraries.

The private library is a partitioned data set (PDS) on a direct access device. Each member is an executable, user-written program.

Syntax:

```
//JOBLIB DD parameter[,parameter]...  [comments]
```

Parameters on JOBLIB DD Statements

When Retrieving a Cataloged Library:

- Code the DSNAME parameter.
- Code the DISP parameter. The status subparameter must be OLD or SHR. The disposition subparameters should indicate what you want done with the private library after its use in the job.
- Do not code VOLUME or UNIT.

When Retrieving a Library that is not Cataloged:

- Code the DSNAME parameter.
- Code the DISP parameter. The DISP parameter must be DISP=(OLD,PASS) or DISP=(SHR,PASS). SHR indicates that the data set is old, but allows other jobs to use the library.
- Code the UNIT parameter.
- Code the VOLUME parameter.

When Creating a Library:

- Code the DSNAME parameter to assign the library a name.
- Code the UNIT parameter. The library must be allocated to a direct access device.
- Code a VOLUME parameter, unless a nonspecific request is to be made for any volume.
- Code the SPACE parameter, allowing enough space for the entire library on one direct access volume. Specify space for the PDS directory.

- Code a DISP parameter. The status is NEW. Code CATLG as the disposition, if you intend to keep the library you are creating. Code PASS as the disposition, if you wish the library to be available throughout the job, but deleted at job termination. Note that you must code a disposition; otherwise, the system assumes DELETE and deletes the library at the end of the first step.

Note: Do not use VSAM for a JOBLIB library.

When Adding Members to the Library:

- In the DSNAME parameter, follow the library name with the name of the program being added to the library. For example, DSNAME=LIBRARY(PROGRAM).

- Code the status in the DISP parameter as MOD. If you cataloged the library when you created it, do not code a disposition. Otherwise, code PASS or CATLG.

- If the JOBLIB library is being created in the job, the JOBLIB DD DISP specified CATLG, and the first step adds a member to it, supply unit and volume information in the first step by coding: VOLUME=REF=*.JOBLIB. This parameter is needed because the library is not actually cataloged until the first step completes execution. Otherwise, unit and volume information should not be supplied for a cataloged library.

- Do not code a SPACE parameter. The JOBLIB DD statement requests space for the entire library.

Other Parameters: Code the DCB parameter if complete data control block information is not contained in the data set label. Do not specify FREE=CLOSE; CLOSE is ignored.

Relationship to Other Control Statements

Concatenating Job Libraries: To specify more than one private library for a job:

- Code a JOBLIB DD statement.

- Immediately follow this statement with DD statements that define other private libraries. Omit a ddname from these subsequent DD statements.

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

Overriding a JOBLIB: If you want the system to ignore the JOBLIB for a particular job step and the step does not require another private library, define the system library on a STEPLIB DD statement. For example, specify:

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

For this particular job step, the system will search SYS1.LINKLIB, as specified on the STEPLIB DD statement, for the program requested in the EXEC statement. The system will not search the JOBLIB.

EXEC Statement COND Parameter: If COND=ONLY is specified on the EXEC statement of a job step and a JOBLIB DD statement is being used, the system does not pass the unit and volume information to any succeeding steps, and the system must search the catalog for the JOBLIB data set's unit and volume information.

Location in the JCL

- The JOBLIB DD statement must immediately follow the JOB statement and any JES statements. There must be no intervening EXEC or other DD statements between the JOBLIB DD statement and the JOB statement.
- If libraries are concatenated to the JOBLIB library, the concatenated DD statements must immediately follow the JOBLIB DD statement.
- Do not include a JOBLIB DD statement in an in-stream or cataloged procedure.

Relationship of a JOBLIB to a STEPLIB

Use a STEPLIB DD statement to define a private library for one job step in a job. If you include a STEPLIB DD statement for a job step and a JOBLIB DD statement for the entire job, the system first searches the step library and then the system library for the requested program. The system ignores the job library for a step that has a STEPLIB DD statement.

Examples of the JOBLIB DD Statement

```
//PAYROLL JOB  JONES,CLASS=C
//JOBLIB  DD   DSNAME=PRIVATE.LIB4,DISP=(OLD,PASS)
//STEP1   EXEC PGM=SCAN
//STEP2   EXEC PGM=UPDATE
//DD1     DD   DSNAME=*.JOBLIB,DISP=(OLD,PASS)
```

The private library requested on the JOBLIB DD statement is cataloged. The system passes catalog information to subsequent job steps. The system searches for the programs SCAN and UPDATE first in PRIVATE.LIB4, then in SYS1.LINKLIB. DD statement DD1 refers to the private library requested in the JOBLIB DD statement.

```
//PAYROLL JOB  FOWLER,CLASS=L
//JOBLIB  DD   DSNAME=PRIV.DEPT58,DISP=(OLD,PASS),
//             UNIT=3350,VOLUME=SER=D58PVL
//STEP    EXEC PGM=DAY
//STEP2   EXEC PGM=BENEFITS
//DD1     DD   DSNAME=*.JOBLIB,VOLUME=REF=*.JOBLIB,
//             DISP=(OLD,PASS)
```

The private library requested on the JOBLIB DD statement is not cataloged; therefore, unit and volume information is specified. The system searches for the programs DAY and BENEFITS first in PRIV.DEPT58, then in SYS1.LINKLIB. DD statement DD1 refers to the private library requested in the JOBLIB DD statement.

```
//TYPE    JOB  MSGLEVEL=(1,1)
//JOBLIB  DD   DSNAME=GROUP8.LEVEL5,DISP=(NEW,CATLG),
//             UNIT=3350,VOLUME=SER=148562,
//             SPACE=(CYL,(50,3,4))
//STEP1   EXEC PGM=DISC
//DDA     DD   DSNAME=GROUP8.LEVEL5(RATE),DISP=MOD,
//             VOLUME=REF=*.JOBLIB
//STEP2   EXEC PGM=RATE
```

The private library requested on the JOBLIB DD statement does not exist yet; therefore, the JOBLIB DD statement contains all the parameters required to define the library. The library is created in STEP1, when DD statement DDA defines the new member RATE for the library. Therefore, the system searches SYS1.LINKLIB for the program named DISC. In STEP2, the system searches for the program RATE first in GROUP8.LEVEL5.

```
//PAYROLL JOB  BIRDSALL,TIME=1440
//JOBLIB  DD   DSNAME=KRG.LIB12,DISP=(OLD,PASS)
//        DD   DSNAME=GROUP31.TEST,DISP=(OLD,PASS)
//        DD   DSNAME=PGMSLIB,UNIT=3350,
//             DISP=(OLD,PASS),VOLUME=SER=34568
```

The three DD statements concatenate the three private libraries. The system searches the libraries for each program in this order:

KRG.LIB12
GROUP31.TEST
PGMSLIB
SYS1.LINKLIB

STEPCAT DD Statement

Purpose: Use the STEPCAT DD statement to define a private VSAM or integrated catalog facility user catalog for the duration of a job step. The system searches the private catalog for data sets before it searches the master catalog or a private catalog associated with the first qualifier of a data set's name.

Do not use the STEPCAT DD statement in a job step that references an SMS-managed data set. SMS only accesses SMS-managed data sets that are cataloged in a system catalog.

You cannot specify OS CVOLs as STEPCAT. Access to an OS CVOL is possible only with a special CVOL pointer in the master catalog.

References: For more information on VSAM data sets, see *VSAM Administration Guide.*

Syntax:

```
//STEPCAT DD DISP={OLD|SHR},DSNAME=private-catalog-name[,parameter]...[comments]
```

Parameters on STEPCAT DD Statements

Do not specify any unit or volume information. The system obtains the location of the private catalog from the master catalog. Do not specify FREE=CLOSE; CLOSE is ignored.

Relationship to Other Control Statements

Concatenating Step Catalogs: To specify more than one private catalog for a step:

- Code a STEPCAT DD statement.
- Immediately follow this statement with DD statements that define other private catalogs. Omit a ddname from these subsequent DD statements.

Overriding a JOBCAT: To override a JOBCAT private catalog with the master catalog for a particular job step, code the following in the job step:

```
//STEPCAT  DD  DISP=OLD,DSNAME=master-catalog-name
```

Location in the JCL

Place a STEPCAT DD statement in any position among the DD statements for a step.

Example of the STEPCAT DD Statement

```
//          EXEC PROC=SNZ12
//STEPCAT   DD   DSNAME=BETTGER,DISP=SHR
```

The STEPCAT DD statement specifies a private catalog that the system uses for this job step only.

STEPLIB DD Statement

Purpose: Use the STEPLIB DD statement to:

- Create a private library.
- Identify a private library that the system is to search for the program named in the EXEC statement PGM parameter. If the system does not find the program in the private library, only then does the system search the system libraries.

The private library is a partitioned data set (PDS) on a direct access device. Each member is an executable, user-written program.

Subsequent job steps in the same job may refer to or receive a private library defined on a STEPLIB DD statement. Also, you can place a STEPLIB DD statement in an in-stream or cataloged procedure.

Syntax:

```
//STEPLIB DD parameter[,parameter]...  [comments]
```

Parameters on STEPLIB DD Statements

When Retrieving a Cataloged Library:

- Code the DSNAME parameter.
- Code the DISP parameter. The status subparameter must be OLD or SHR. The disposition subparameters should indicate what you want done with the private library after its use in the job step.
- Do not code VOLUME or UNIT.

When Retrieving a Library Passed from a Previous Step: In the passing job step, code a DISP disposition subparameter of PASS when a step library is to be used by subsequent steps in the job.

In a receiving step:

- Code in the DSNAME parameter either the name of the step library or a backward reference of the form *.stepname.STEPLIB. If the step library is defined in a procedure, the backward reference must include the procedure step name: *.stepname.procstepname.STEPLIB.
- Code the DISP parameter. The status subparameter must be OLD. The disposition subparameters should indicate what you want done with the private library after its use in the receiving step.

When Retrieving a Library that is Neither Cataloged Nor Passed:

- Code the DSNAME parameter.
- Code the DISP parameter. The status subparameter must be OLD or SHR. The disposition subparameters should indicate what you want done with the private library after its use in the job step.
- Code the UNIT parameter.
- Code the VOLUME parameter.

When Creating a Library:

- Code the DSNAME parameter to assign the library a name.
- Code the UNIT parameter. The library must be allocated to a direct access device.
- Code a VOLUME parameter, unless a nonspecific request is to be made for any volume.
- Code the SPACE parameter, allowing enough space for the entire library on one direct access volume. Specify space for the PDS directory.
- Code a DISP parameter. The status is NEW. Code CATLG as the disposition, if you intend to keep the library you are creating. Code PASS as the disposition, if you wish the library to be available to a following step. Note that you must code a disposition; otherwise, the system assumes DELETE and deletes the library at the end of the step.

Note: Do not use VSAM for a STEPLIB library.

When Adding Members to the Library:

- In the DSNAME parameter, follow the library name with the name of the program being added to the library. For example, DSNAME=LIBRARY(PROGRAM).
- Code the status in the DISP parameter as MOD. If the library is cataloged, do not code a disposition. Otherwise, code PASS or CATLG.
- If the library is cataloged, do not code unit and volume information. Otherwise, code UNIT and VOLUME.
- Do not code a SPACE parameter. The STEPLIB DD statement requests space for the entire library.

Other Parameters: Code the DCB parameter if complete data control block information is not contained in the data set label. Do not specify FREE=CLOSE; CLOSE is ignored.

STEPLIB DD

Relationship to Other Control Statements

Concatenating Step Libraries: To specify more than one private library for a step:

- Code a STEPLIB DD statement.
- Immediately follow this statement with DD statements that define other private libraries. Omit a ddname from these subsequent DD statements.

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

Overriding a JOBLIB: If you want the system to ignore the JOBLIB for a particular job step and the step does not require another private library, define the system library on a STEPLIB DD statement. For example, specify:

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

For this particular job step, the system will first search SYS1.LINKLIB, as specified on the STEPLIB DD statement, for the program requested in the EXEC statement. The system will not search the JOBLIB.

Location in the JCL

Place a STEPLIB DD statement in any position among the DD statements for a step.

If libraries are concatenated to the STEPLIB library, the concatenated DD statements must immediately follow the STEPLIB DD statement.

Relationship of a STEPLIB to a JOBLIB

Use a JOBLIB DD statement to define a private library that the system is to use for an entire job. If you include a JOBLIB DD statement for the job and a STEPLIB DD statement for an individual job step, the system first searches the step library and then the system library for the program requested in the EXEC statement. The system ignores the JOBLIB library for that step.

Examples of the STEPLIB DD Statement

```
//PAYROLL JOB  BROWN,MSGLEVEL=1
//STEP1   EXEC PROC=LAB14
//STEP2   EXEC PGM=SPKCH
//STEPLIB DD   DSNAME=PRIV.LIB5,DISP=(OLD,KEEP)
//STEP3   EXEC PGM=TIL80
//STEPLIB DD   DSNAME=PRIV.LIB12,DISP=(OLD,KEEP)
```

The system searches PRIV.LIB5 for the program SPKCH and PRIV.LIB12 for TIL80. The system catalogs both private libraries.

```
//PAYROLL JOB  BAKER,MSGLEVEL=1
//JOBLIB  DD   DSNAME=LIB5.GROUP4,DISP=(OLD,PASS)
//STEP1   EXEC PROC=SNZ12
//STEP2   EXEC PGM=SNAP10
//STEPLIB DD   DSNAME=LIBRARYP,DISP=(OLD,PASS),
//             UNIT=3350,VOLUME=SER=55566
//STEP3   EXEC PGM=A1530
//STEP4   EXEC PGM=SNAP11
//STEPLIB DD   DSNAME=*.STEP2.STEPLIB,
//             DISP=(OLD,KEEP)
```

The system searches LIBRARYP for program SNAP10; LIBRARYP is passed to subsequent steps of this job. The STEPLIB DD statement in STEP4 refers to the LIBRARYP library defined in STEP2; the system searches LIBRARYP for SNAP11. Since a JOBLIB DD statement is included, the system searches for programs SNZ12 and A1530 first in LIB5.GROUP4, then in SYS1.LINKLIB.

```
//PAYROLL JOB  THORNTON,MSGLEVEL=1
//JOBLIB  DD   DSNAME=LIB5.GROUP4,DISP=(OLD,PASS)
//STEP1   EXEC PGM=SUM
//STEPLIB DD   DSNAME=SYS1.LINKLIB,DISP=OLD
//STEP2   EXEC PGM=VARY
//STEP3   EXEC PGM=CALC
//STEPLIB DD   DSNAME=PRIV.WORK,DISP=(OLD,PASS)
//         DD   DSNAME=LIBRARYA,DISP=(OLD,KEEP),
//             UNIT=3350,VOLUME=SER=44455
//         DD   DSNAME=LIB.DEPT88,DISP=(OLD,KEEP)
//STEP4   EXEC PGM=SHORE
```

For STEP2 and STEP4, the system searches the private library named LIB5.GROUP4 defined in the JOBLIB DD statement first for programs VARY and SHORE. For STEP1, the system searches SYS1.LINKLIB first for program SUM, because the STEPLIB DD statement names the system library.

A concatenation of private libraries is defined in STEP3. The system searches for the program named CALC in this order: PRIV.WORK, LIBRARYA, LIB.DEPT88, SYS1.LINKLIB. If a later job step refers to the STEPLIB DD statement in STEP3, the system will search for the program in the private library named PRIV.WORK and, if it is not found there, in SYS1.LINKLIB; the concatenated libraries are not searched.

SYSABEND, SYSMDUMP, and SYSUDUMP DD Statements

Purpose: Use a SYSABEND, SYSMDUMP, or SYSUDUMP DD statement in a job step to direct the system to produce a dump. The system produces the requested dump:

- If the step terminates abnormally.
- If the step starts to terminate abnormally, but system recovery procedures enable the step to terminate normally.

The dump DD statements for requesting dumps are:

SYSABEND DD statement
Produces a dump of user and system areas; this dump contains all the areas dumped in a SYSUDUMP plus the local system queue area (LSQA), including subpools 229 and 230, and the input/output system (IOS) control blocks for the failing task. The dump is formatted, so that it can be printed directly.

SYSMDUMP DD statement
Produces a dump of the system areas and the program's address space. The dump is unformatted and machine-readable; to be used, it must be printed by the PRDUMP service aid or by the interactive problem control system (IPCS).

SYSUDUMP DD statement
Produces a dump of user areas. The dump is formatted, so that it can be printed directly.

The dump contents are as described only when the installation uses the IBM-supplied defaults for the dumps. The contents of these dumps can be set during system initialization and/or can be changed for an individual dump in the ABEND macro instruction, in a CHNGDUMP command, and by a SLIP command. For details, see *SPL: Initialization and Tuning*.

Dumps are optional; use a dump DD statement only when you want to produce a dump.

References: For information on how to interpret dumps, see *Debugging Handbook* and *Diagnostic Techniques*.

Syntax:

```
//SYSABEND  DD parameter[,parameter]...  [comments]
//SYSMDUMP  DD parameter[,parameter]...  [comments]
//SYSUDUMP  DD parameter[,parameter]...  [comments]
```

Location in the JCL

Do not place in the same job step two DD statements with the same dump ddname.

Storing a Dump

If you wish to store a dump instead of having it printed, code the following parameters on the dump DD statement:

- The DSNAME parameter.
- The UNIT parameter.
- The VOLUME parameter.
- The DISP parameter. The data set's status is NEW. Because you want to store the data set, make the data set's abnormal termination disposition KEEP or CATLG.
- The SPACE parameter, if the dump is written on direct access.

Note: Do not use VSAM for dump data sets.

SYSMDUMP Requirements: The SYSMDUMP DD statement must specify a magnetic tape unit or a direct access device.

To write more than one SYSMDUMP dump in the same data set on tape, specify the following:

- DSNAME=SYS1.SYSMDPxx where xx is 00 through FF. SYSMDPxx is a preallocated data set that you must initialize with an end-of-file (EOF) mark on the first record.
- DISP=SHR

You can ask the system to write additional dumps only if you off-load any previous dump and write an EOF mark at the beginning of the SYS1.SYSMDPxx data set. To accomplish this, your installation must install an exit routine for message IEA993. For information on this exit routine, see *SPL: User Exits*.

Printing a Dump

To print a dump for either a SYSABEND or SYSUDUMP DD statement, code one of the following on the DD statement for the output data set:

- A UNIT parameter that specifies a printer.
- The SYSOUT parameter that specifies a print output class.

To print a dump for a SYSMDUMP DD statement, use one of the following programs:

PRDMP service aid
This program is described in *SPL: Service Aids*. When using PRDMP, the SYSMDUMP DD statement must allocate the dump data set to a magnetic tape or a direct access device.

IPCS
This program is described in *Interactive Problem Control System (IPCS) User's Guide and Reference*. When using IPCS, the data set disposition affects the collection of events.

If you print the dump in a JES3 system on a 3800 Printing Subsystem, code CHARS=DUMP for a dump with 204 characters per line and FCB=STD3 for 8 lines per inch.

Overriding Dump DD Statements

To change the type of dump requested in a dump DD statement in a cataloged or in-stream procedure, the ddname of the overriding DD statement in the calling step must be different from the dump ddname of the procedure DD statement.

Duplicate Dump Requests

You can code more than one dump request in a job step using DD statements that have **different** ddnames. When you do this, the system uses the last dump DD statement it encounters.

When the system finds dump DD statements with duplicate ddnames, processing is as follows:

- **In a JES2 system**, the job fails with message IEA912I.
- **In a JES3 system**:
  - If both DD statements request JES3- or jointly-managed devices, the job is cancelled during JES3 interpretation.
  - If only one or neither statement requests JES3- or jointly-managed devices, the job fails with message IEA912I.

Examples of the SYSABEND, SYSMDUMP, and SYSUDUMP DD Statements

```
//STEP2    EXEC PGM=A
//SYSUDUMP DD   SYSOUT=A
```

The SYSUDUMP DD statement specifies that you want the dump routed to system output class A.

```
//SYSMDUMP DD  DSNAME=DUMP,DISP=(NEW,KEEP),
//             UNIT=3400-6,VOLUME=SER=147958
```

The SYSMDUMP DD statement specifies that the dump is to be stored on a tape. Because the LABEL parameter is not coded, the tape must have IBM standard labels.

```
//STEP1    EXEC PGM=PROGRAM1
//SYSABEND DD   DSNAME=DUMP,UNIT=3350,DISP=(,PASS,KEEP),
//             VOLUME=SER=1234,SPACE=(TRK,(40,20))
//STEP2    EXEC PGM=PROGRAM2
//SYSABEND DD   DSNAME=*.STEP1.SYSABEND,DISP=(OLD,DELETE,KEEP)
```

Both SYSABEND DD statements specify that the dump is to be stored. The space request in STEP1 is ample and will not inhibit dumping due to insufficient space. If STEP1 does not abnormally terminate but STEP2 does, the system writes the dump for STEP2 in the space allocated in STEP1. In both steps, an abnormal termination disposition of KEEP is specified so that the dump is stored if either of the steps abnormally terminates. If both of the steps successfully execute, the second DISP subparameter, DELETE, in STEP2 instructs the system to delete the data set and free the space acquired for dumping.

```
//STEP1    EXEC PGM=WWK
//SYSMDUMP DD   DSNAME=DUMP,UNIT=3350,DISP=(,DELETE,
//           KEEP),VOLUME=SER=54366,SPACE=(1680,(160,80))
//STEP2    EXEC PGM=IKJEFT01,PARM=AMDPRDMP,COND=ONLY
//IN       DD   DSNAME=*.STEP1.SYSUDUMP,DISP=(OLD,DELETE),
//           VOLUME=REF=*.STEP1.SYSUDUMP
```

STEP1 specifies that the dump is to be stored on a direct access device if the step abnormally terminates. Because the EXEC COND=ONLY parameter is specified in STEP2, STEP2 is executed only if STEP1 abnormally terminates. STEP2 executes the PRDMP service aid to print the dump.

```
//STEP     EXEC PGM=EXSYSM
//SYSMDUMP DD   UNIT=3330,VOLUME=SER=123456,SPACE=(CYL,(0,1)),
//           DISP=(NEW,DELETE,KEEP),DSNAME=MDUMP
```

The SYSMDUMP DD statement allocates dump data set MDUMP to a direct access device.

```
//STEP     EXEC PGM=EXSYSMDP
//SYSMDUMP DD   DSNAME=SYS1.SYSMDP00,DISP=SHR
```

The SYSMDUMP is written in data set SYS1.SYSMDP00.

Note: When you specify a DSNAME of SYS1.SYSMDPxx and DISP=SHR, the first SYSMDUMP produced is retained on the data set. This first SYSMDUMP must be off-loaded and an end-of-file mark written at the beginning of the SYS1.SYSMDPxx data set before subsequent dumps can be written.

```
//STEPA    EXEC PGM=EXSYSM2,ADDRSPC=REAL
//SYSMDUMP DD   UNIT=SYSDA,SPACE=(CYL,(0,1)),
//           DISP=(NEW,PASS)
//STEP2    EXEC PGM=IKJEFT01,PARM=AMDPRDMP,COND=ONLY
//SYSUT1   DD   DSNAME=*.STEPA.SYSMDUMP,DISP=OLD
//PRINTER  DD   SYSOUT=A
//SYSIN    DD   *
         FORMAT
         LOGDATA
         END
/*
```

In STEPA, the SYSMDUMP DD statement directs output to a VIO data set (1) by specifying a VIO-eligible device group (SYSDA) and (2) by not assigning a data set name to make the data set temporary and eligible for VIO. STEPB is executed only if STEPA abnormally terminates. In STEPB, the dump output is printed by the PRDMP service aid on a device assigned to output class A.

SYSCHK DD Statement

Purpose: Use the SYSCHK DD statement to define a checkpoint data set that the system is to write during execution of a processing program. Use this statement again when the step is restarted from a checkpoint written in the data set.

Note: If restart is to begin at a step, as indicated by the RD parameter on the EXEC statement, do not use a SYSCHK DD statement.

References For detailed information about the checkpoint/restart facilities, see *Checkpoint/Restart User's Guide.*

Syntax:

```
//SYSCHK DD parameter[,parameter]...  [comments]
```

Parameters on SYSCHK DD Statements

When Creating a Checkpoint Data Set:

- Code a SPACE parameter, but do not request secondary space.
  - The **primary space** request must be large enough to hold all checkpoints. Although your program or the system can write checkpoints in secondary space, the system **cannot** perform a restart from checkpoints in secondary space.
  - If you do **not** request **secondary space** and the primary space fills up, the job abnormally terminates. You can successfully restart the job at the last checkpoint; however, when the processing program or system writes the next checkpoint the job abnormally terminates again.
  - If you **do** request **secondary space** and the primary space fills up, the processing program or the system writes one invalid checkpoint followed by successful checkpoints. An attempt to restart from one of the checkpoints following the invalid checkpoint results in abnormal termination.
- Code the RLSE subparameter of the SPACE parameter only if the processing program opens the checkpoint data set and the checkpoint data set remains open until the end of the program. If you specify RLSE, the system releases unused space after the first CLOSE macro instruction.

  Do **not** code the RLSE subparameter:
  - If the processing program opens the checkpoint data set before writing each checkpoint and closes the checkpoint data set after writing each checkpoint. The system releases all unused space while closing the data set after the first checkpoint, leaving no space for additional checkpoints.
  - If the system opens the checkpoint data set. The system opens and closes the checkpoint data set before it writes the first checkpoint. With RLSE specified, the system would release all space before the first checkpoint could be written.

- Code the **CONTIG** subparameter of the SPACE parameter to request contiguous space. The system otherwise provides additional primary space using extents. If the extents are **not** contiguous, any checkpoints in these extents cannot be used for a successful restart.

When Retrieving a Cataloged Checkpoint Data Set:

- Code the DSNAME parameter.
- Code the DISP parameter to specify a status of OLD and a disposition of KEEP.
- Code the VOLUME parameter. If the checkpoint entry is on a tape volume other than the first volume of the checkpoint data set, code the volume serial number or volume sequence number to identify the correct volume. The serial number of the volume on which a checkpoint entry was written appears in the console message issued after the checkpoint entry is written.
- Code the UNIT parameter, if you coded the VOLUME parameter, because the system will not look in the catalog for unit information.

When Retrieving a Checkpoint Data Set that is not Cataloged:

- Code the DSNAME parameter. If the checkpoint data set is partitioned, do not code a member-name in the DSNAME parameter.
- Code the DISP parameter to specify a status of OLD and a disposition of KEEP.
- Code the VOLUME parameter. The serial number of the volume on which a checkpoint entry was written appears in the console message issued after the checkpoint entry is written.
- Code the UNIT parameter.

Other Parameters:

- Code the LABEL parameter if the checkpoint data set does not have standard labels.
- Code DCB=TRTCH=C if the checkpoint data set is on 7-track magnetic tape with nonstandard labels or no labels.
- If the volume containing the checkpoint data set is to be mounted on a JES3-managed device, do not code the DEFER subparameter of the UNIT parameter on the SYSCHK DD statement.

Note: Do not use VSAM for checkpoint data sets.

Relationship to Other Control Statements

Code the RESTART parameter on the JOB statement; without it, the system ignores the SYSCHK DD statement.

SYSCHK DD

Location in the JCL

- When writing checkpoints, place the SYSCHK DD statement after any JOBLIB DD statements, if coded; otherwise, after the JOB statement.
- When restarting a job from a checkpoint, place the SYSCHK DD statement immediately before the first EXEC statement of the resubmitted job.

Examples of the SYSCHK DD Statement

```
//JOB1   JOB  RESTART=(STEP3,CK3)
//SYSCHK DD   DSNAME=CHLIB,UNIT=3350,
//            DISP=OLD,VOLUME=SER=456789
//STEP1  EXEC PGM=A
```

The checkpoint data set defined on the SYSCHK DD statement is not cataloged.

```
//JOB2   JOB  RESTART=(STEP2,NOTE2)
//JOBLIB DD   DSNAME=PRIV.LIB3,DISP=(OLD,PASS)
//SYSCHK DD   DSNAME=CHECKPTS,DISP=(OLD,KEEP),
//            UNIT=3400-6,VOLUME=SER=438291
//STEP1  EXEC PGM=B
```

The checkpoint data set defined on the SYSCHK DD statement is not cataloged. Note that the SYSCHK DD statement follows the JOBLIB DD statement.

SYSCKEOV DD Statement

Purpose: Use the SYSCKEOV DD statement to define a checkpoint data set for checkpoint records from the checkpoint at end-of-volume (EOV) facility. The checkpoint at EOV facility is invoked by a DD CHKPT parameter.

References: For information on the DD CHKPT parameter, see "CHKPT Parameter" on page 10-31. For information on checkpoint/restart facilities, see *Checkpoint/Restart User's Guide*.

Syntax:

```
//SYSCKEOV DD parameter[,parameter]...  [comments]
```

Parameters on SYSCKEOV DD Statements

When Creating a Checkpoint Data Set:

- Code a SPACE parameter, but do not request secondary space. The **primary space** request must be large enough to hold all checkpoints; if not, the job abnormally terminates.
- Do not code the RLSE subparameter of the SPACE parameter.
- Code the CONTIG subparameter of the SPACE parameter to request contiguous space. The system otherwise provides additional primary space using extents.
- The SYSCKEOV DD statement must define a BSAM data set.
- Code DISP=MOD to reduce loss of checkpoint data in case of a system failure during checkpointing.

Other Parameters:

- Do not code on the SYSCKEOV DD statement the following:
  - CHKPT=EOV parameter.
  - DCB parameter. All DCB information is provided by the checkpoint at EOV facility.
  - DEFER subparameter of the UNIT parameter.
- If you code the LABEL parameter, you must specify LABEL=(,SL) for IBM standard labels.
- If the SYSCKEOV data set resides on a direct access storage device, that device cannot be shared with another processor.

SYSCKEOV DD

Location in the JCL

If you code a CHKPT parameter on any DD statements in a job step, place a SYSCKEOV DD statement in the DD statements for the step.

Example of the SYSCKEOV DD Statement

```
//SYSCKEOV DD  DSNAME=CKPTDS,UNIT=TAPE,DISP=MOD
```

This statement defines a checkpoint data set for checkpoint at EOV records.

SYSIN DD Statement

Purpose: Use a SYSIN DD statement to begin an in-stream data set. In-stream data sets begin with a DD * or DD DATA statement; these DD statements can have any valid ddname, including SYSIN. If you omit a DD statement before input data, the system provides a DD * statement with the ddname of SYSIN.

Syntax:

```
//SYSIN DD parameter[,parameter]...  [comments]
```

Parameters on SYSIN DD Statements

The first parameter is an * or DATA, to signal that an in-stream data set follows immediately.

Location in the JCL

A SYSIN DD statement appears at the beginning of an in-stream data set.

Examples of SYSIN DD Statements

```
//STEP1  EXEC  PGM=READ
//SYSIN  DD    *
         .
         .
         data
         .
//OUT1   DD    SYSOUT=A
//STEP2  EXEC  PGM=WRITE
//SYSIN  DD    DATA,DLM=17
         .
         .
         .
17
```

Chapter 12. Delimiter Statement

Purpose: Use the delimiter statement to indicate the end of data or transmittal records in the input stream.

Syntax:

```
/* [comments]
xx [comments]
```

A delimiter statement consists of the characters /* or the two characters specified in a DLM parameter in columns 1 and 2 and one field: comments.

Do not continue a delimiter statement.

Comments Field

The comments field follows the delimiter characters.

For JES2, code any comments in columns 4 through 80. (A blank must follow the delimiter characters.)

For JES3, code any comments in columns 3 through 80.

Relationship to the DLM Parameter

The system recognizes a delimiter other than /* if a DLM parameter is coded on:

- The DD * or DD DATA statement that defines an in-stream data set.
- The XMIT JCL statement that precedes input stream records to be transmitted to another node.
- The JES2 /*XMIT statement that precedes input stream records to be transmitted to another node.

A delimiter statement is optional:

- If the data is preceded by a DD * statement without a DLM parameter.
- If transmitted records are preceded by an /*XMIT statement without a DLM parameter.

Delimiter Statement

Location in the JCL

A delimiter statement must appear:

- At the end of an in-stream data set that begins with a DD DATA statement.
- At the end of an in-stream data set that begins with a DD statement containing a DLM parameter.
- At the end of records to be transmitted to another node when the records are preceded by an /*XMIT statement containing a DLM parameter.
- At the end of records to be transmitted to another node when the records are preceded by an XMIT JCL statement.

Examples of the Delimiter Statement

```
//JOB54   JOB   ,'C BROWN',MSGLEVEL=(2,0)
//STEPA   EXEC  PGM=SERS
//DD1     DD    *
          .
          .
          data
          .
/*        END OF DATA FOR DATA SET DD1
//DD2     DD    DATA,DLM=AA
          .
          .
          data
          .
AA        END OF DATA FOR DATA SET DD2
```

```
//JOB54   JOB   ,'C BROWN',MSGLEVEL=(2,0)
//        XMIT  DEST=NODEA,DLM=BB
//JOB55   JOB   ,'C BROWN',MSGLEVEL=(2,0)
//STEPA   EXEC  PGM=SERS
//DD1     DD    *
          .
          .
          data
          .
/*        END OF DATA FOR DATA SET DD1
//DD2     DD    DATA,DLM=AA
          .
          .
          data
          .
AA        END OF DATA FOR DATA SET DD2
BB        END OF TRANSMITTED JOB
```

This example shows nested delimiter statements.

Chapter 13. ENDCNTL Statement

Purpose: Use the ENDCNTL statement to mark the end of the program control statements following a CNTL statement.

Syntax:

```
//[label] ENDCNTL [comments]
```

The ENDCNTL statement consists of the characters // in columns 1 and 2 and three fields: label, operation (ENDCNTL), and comments.

Label Field

Code a label on the ENDCNTL statement, as follows:

- Each label must be unique within the job.
- The label must begin in column 3.
- The label is 1 through 8 alphanumeric or national ($, #, @) characters.
- The first character must be alphabetic or national ($, #, @)
- The label must be followed by at least one blank.

Operation Field

The operation field consists of the characters ENDCNTL and must be preceded and followed by at least one blank. It can begin in any column.

Comments Field

The comments field follows the ENDCNTL after at least one intervening blank.

Location in the JCL

The ENDCNTL statement immediately follows the one or more program control statements following a CNTL statement. Thus, the ENDCNTL statement can appear in a job step or in a cataloged or in-stream procedure.

ENDCNTL

Example of the ENDCNTL Statement

```
//STEP1     EXEC  PGM=PRINT
//ABLE      CNTL
//STATE1    PRINTDEV BUFNO=20,PIMSG=YES,DATACK=BLOCK
//BAKER     ENDCNTL
//CALLER    DD    UNIT=3800-3,CNTL=*.ABLE
```

Chapter 14. EXEC Statement

Purpose: Use the EXEC (execute) statement to identify the program or cataloged or in-stream procedure that this job step is to execute and to tell the system how to process the job step. The EXEC statement marks the beginning of each step in a job or a procedure.

A job can have a maximum of 255 job steps. This maximum includes all steps in any procedures the EXEC statements call.

The parameters you can specify for step processing are arranged alphabetically in the following pages.

References: For information about the JES initialization parameters that provide installation defaults, see *SPL: JES2 Initialization and Tuning* and *SPL: JES3 Initialization and Tuning*.

Syntax:

```
//[stepname]  EXEC  positional-parameter[,keyword-parameter]...[comments]
```

The EXEC statement consists of the characters // in columns 1 and 2 and four fields: name, operation (EXEC), parameter, and comments.

An EXEC statement is required for each job step.

Name Field

A stepname is optional, but is needed for the following. When a stepname is needed, it must be unique within the job, including stepnames in any procedures called by the job.

- Referring to the step in later job control statements.
- Overriding parameters on an EXEC statement or DD statement in a cataloged or in-stream procedure step.
- Adding DD statements to a cataloged or in-stream procedure step. However, a stepname is not required when adding to the first step in a procedure.
- Performing a step or checkpoint restart at or in the step.
- Identifying a step in a cataloged or in-stream procedure.

EXEC

Code a stepname as follows:

- The stepname must begin in column 3.
- The stepname is 1 through 8 alphanumeric or national ($, #, @) characters.
- The first character must be alphabetic or national ($, #, @).
- The stepname must be followed by at least one blank.

Operation Field

The operation field consists of the characters EXEC and must be preceded and followed by at least one blank. It can begin in any column.

Parameter Field

An EXEC statement has two kinds of parameters: positional and keyword.

Do not use EXEC statement parameter keywords as symbolic parameters, names, or labels.

Positional Parameters: An EXEC statement must contain **one** of the positional parameters: PGM, PROC, or procedure name. This positional parameter must precede all keyword parameters.

POSITIONAL PARAMETERS	VALUES	PURPOSE
PGM = { program-name / *.stepname.ddname / *.stepname.procstepname.ddname / JCLTEST / JSTTEST } See page 14-23	program-name: 1 - 8 alphanumeric or $, #, @ characters ddname: name of DD for PDS member containing program stepname: DD in named step procstepname: step in named procedure JCLTEST and JSTTEST: scan for syntax without executing the job (JES3 only)	Names the program the system is to execute or, for JES3 only, requests syntax check without execution.
{ PROC = procedure-name / procedure-name } See page 14-25	procedure-name: 1 - 8 alphanumeric or $, #, @ characters	Names the cataloged or in-stream procedure the system is to call and execute.

Keyword Parameters: An EXEC statement can contain the following keyword parameters. You can code any of the keyword parameters in any order in the parameter field after the positional parameter.

KEYWORD PARAMETERS	VALUES	PURPOSE
ACCT[.procstepname] = (accounting-information) See page 14-5	accounting-information: up to 142 characters [.procstepname]: name of procedure EXEC containing ACCT to be affected	Specifies accounting information for the step.
ADDRSPC[.procstepname] = { VIRT / REAL } See page 14-7	VIRT: virtual (pageable) storage REAL: real (nonpageable) storage [.procstepname]: name of procedure EXEC containing ADDRSPC to be affected	Indicates the type of storage required for the step.

KEYWORD PARAMETERS	VALUES	PURPOSE
COND[.procstepname]= ((code,operator[,stepname][.procstepname]) [,(code,operator[,stepname][.procstepname])...] [,EVEN ,ONLY]) See page 14-9	code: 0 - 4095 operator: GT Code from GE chart on EQ page 14-13 NE LT LE EVEN: execute step even if preceding step abnormally terminated ONLY: execute step only if preceding step abnormally terminated stepname: step issuing return code procstepname: step is in named procedure [.procstepname]: name of procedure EXEC containing COND to be affected	Specifies the return code tests used to determine if this step is to be executed or bypassed.
DPRTY[.procstepname]=([value1][,value2]) See page 14-15	value1: 0 - 15 value2: 0 - 15 [.procstepname]: name of procedure EXEC containing DPRTY to be affected	Assigns a dispatching priority for the address space: dispatching priority = (value1 * 16) + value2
DYNAMNBR[.procstepname]=n See page 14-17	n: 0 - 3273 minus number of DD statements in step [.procstepname]: name of procedure EXEC containing DYNAMNBR to be affected	Holds a number of data set allocations for reuse.
PARM[.procstepname]=information See page 14-19	information: up to 100 characters [.procstepname]: name of procedure EXEC containing PARM to be affected	Passes variable information to the processing program.
PERFORM[.procstepname]=n See page 14-21	n: 1 - 999 [.procstepname]: name of procedure EXEC containing PERFORM to be affected	Specifies the step's performance group, which determines the rate at which the step has access to the processor, storage, and channels.
RD[.procstepname]={R RNC NR NC} See page 14-26	R: restart, checkpoints allowed RNC: restart, no checkpoints NR: no restart, checkpoints allowed NC: no restart, no checkpoints [.procstepname]: name of procedure EXEC containing RD to be affected	Indicates whether the operator should perform automatic step restart, if the step fails, and controls whether checkpoints are written for CHKPT macros or DD statement CHKPT parameters.
REGION[.procstepname]={valueK valueM} See page 14-30	valueK: even number, 1 - 7 digits from 1 - 2096128 valueM: even or odd number, 1 - 4 digits from 1 - 2047 [.procstepname]: name of procedure EXEC containing REGION to be affected	Specifies the amount of space in kilobytes or megabytes required by the step.
TIME[.procstepname]={([minutes][,seconds]) 1440} See page 14-33	minutes: 1 - 1439 seconds: 1 - 59 [.procstepname]: name of procedure EXEC containing TIME to be affected	Specifies the maximum time the step is to use the processor and requests messages giving the time used.

Keyword Parameters on EXEC Statement that Calls a Procedure: When the EXEC statement positional parameter calls a cataloged or in-stream procedure, all of the EXEC statement's keyword parameters override matching EXEC keyword parameters in the called procedure. If a keyword parameter is to override a parameter on only one EXEC statement in the procedure, code **.procstepname** immediately following the keyword:

```
keyword.procstepname=value
```

The procstepname is the name field on the procedure EXEC statement containing the keyword parameter to be overridden. For example:

```
//STEP1  EXEC  PROC=WKREPORT,ACCT.PSTEPWED=5670
```

The accounting information **5670** applies only to step PSTEPWED in the procedure WKREPORT.

Comments Field

The comments field follows the parameter field after at least one intervening blank.

Location in the JCL

An EXEC statement must be the first statement in each job step or cataloged or in-stream procedure step.

Examples of EXEC Statements

```
//STEP4  EXEC PGM=DREC,PARM='3018,NO'
```

The EXEC statement named STEP4 invokes a program named DREC and passes the value in the PARM parameter to DREC.

```
//       EXEC PGM=ENTRY,TIME=(2,30)
```

This EXEC statement, which does not have a stepname, invokes a program named ENTRY and specifies the maximum processor time for execution of the step.

```
//FOR    EXEC PROC=PROC489,ACCT=DB1528,RD.PSTEP2=RNC,DEV=3350
```

The EXEC statement named FOR invokes a cataloged or in-stream procedure named PROC489. The ACCT parameter applies to all steps in the procedure. The RD parameter applies to only the step named PSTEP2. The DEV parameter assigns the value 3350 to the symbolic parameter &DEV in a procedure statement.

ACCT Parameter

Parameter Type: Keyword, optional

Purpose: Use the ACCT parameter to specify one or more subparameters of accounting information that apply to this step. The system passes the accounting information to the installation's accounting routines.

References: For more information on how to add accounting routines, see *SPL: System Management Facilities.*

Syntax:

```
ACCT[.procstepname]=(accounting-information)
```

Single Subparameter: You can omit the parentheses if the accounting information consists of only one subparameter.

Length: The entire accounting-information must not exceed 142 characters:

- Including any commas, which are considered part of the information.
- Excluding any enclosing parentheses or apostrophes, which are not considered part of the information.

Multiple Subparameters: When the accounting-information consists of more than one subparameter, separate the subparameters by commas and enclose the information in parentheses or apostrophes. For example, ACCT=(5438,GROUP6) or ACCT='5438,GROUP6'.

Special Characters: When a subparameter contains special characters, other than hyphens or plus zero (+0, an overpunch), enclose it in apostrophes and the information in parentheses or enclose all of the information in apostrophes. For example, ACCT=(387,'72/159') or ACCT='387,72/159'.

Code each apostrophe that is part of the accounting-information as two consecutive apostrophes. For example, code DEPT'D58 as ACCT='DEPT''D58'

Continuation onto Another Statement: Enclose the accounting-information in parentheses. End each statement with a comma after a complete subparameter. For example:

```
//STEP1  EXEC  PGM=WRITER,ACCT=(1417,J318,'D58/920','CHG=2',
//    '33.95')
```

EXEC: ACCT

Subparameter Definition

accounting-information
Specifies one or more subparameters of accounting information, as defined by the installation.

On EXEC Statement that Calls a Procedure

If the EXEC statement calls a cataloged or in-stream procedure, the ACCT parameter overrides the ACCT parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The information applies only to the named procedure step. The EXEC statement can have as many ACCT.procstepname parameters as the procedure has steps; each ACCT parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the information applies to all steps in the called procedure.

Examples of the ACCT Parameter

```
//STEP1  EXEC PGM=JP5,ACCT=(LOCATION8,'CHGE+3')
```

This EXEC statement executes program JP5 and specifies accounting information for this job step.

```
//STP3  EXEC PROC=LOOKUP,ACCT=('/83468')
```

This EXEC statement calls cataloged or in-stream procedure LOOKUP. The accounting information applies to this job step, STP3, and to all the steps in procedure LOOKUP.

```
//STP4  EXEC PROC=BILLING,ACCT.PAID=56370,ACCT.LATE=56470,
//      ACCT.BILL='121+366'
```

This EXEC statement calls cataloged or in-stream procedure BILLING. The statement specifies different accounting information for each of the procedure steps: PAID, LATE, and BILL.

ADDRSPC Parameter

Parameter Type: Keyword, optional

Purpose: Use the ADDRSPC parameter to indicate to the system that the job step requires virtual storage (pageable) or real storage (nonpageable).

Syntax:

```
ADDRSPC[.procstepname]={VIRT}
                       {REAL}
```

Subparameter Definition

VIRT
Requests virtual storage. The system **can** page the job step.

REAL
Requests real storage. The system **cannot** page the job step and must place the job step in real storage.

Defaults

If no ADDRSPC parameter is specified, the default is VIRT.

Overrides

The JOB statement ADDRSPC parameter applies to all steps of the job and overrides any EXEC statement ADDRSPC parameters.

Code EXEC statement ADDRSPC parameters when each job step requires different types of storage. The system uses an EXEC statement ADDRSPC parameter only when no ADDRSPC parameter is on the JOB statement and only during the job step.

Relationship to the EXEC REGION Parameter

When ADDRSPC=REAL: Code a REGION parameter to specify how much real storage the job needs. If you omit the REGION parameter, the system uses the default.

When ADDRSPC=VIRT or ADDRSPC is Omitted: Code a REGION parameter to specify how much virtual storage the job needs. If you omit the REGION parameter, the system uses the default.

EXEC: ADDRSPC

On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the ADDRSPC parameter overrides the ADDRSPC parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The parameter applies only to the named procedure step. The EXEC statement can have as many ADDRSPC.procstepname parameters as the procedure has steps; each ADDRSPC parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to all steps in the called procedure.

Examples of the ADDRSPC Parameter

```
//CAC1  EXEC PGM=A,ADDRSPC=VIRT
```

This EXEC statement executes program A and requests virtual (pageable) storage. Because the REGION parameter is not specified, the storage available to this job step is the installation default or the region size specified on the JOB statement.

```
//CAC2  EXEC PROC=B,ADDRSPC=REAL,REGION=80K
```

This EXEC statement calls procedure B and requests real (nonpageable) storage. The REGION parameter specifies 80K of storage.

COND Parameter

Parameter Type: Keyword, optional

Purpose: Use the COND parameter to specify the return code tests the system is to use to determine whether to bypass or execute this job step. The system performs each COND parameter test against the return code from every previous job step that executed or from the named previous job step(s). If none of these tests is satisfied, the system executes this job step; if any test is satisfied, the system bypasses this job step.

Bypassing a step because of a return code test is not the same as abnormally terminating the step. The system abnormally terminates a step following an error so serious that it prevents successful execution. In contrast, bypassing of a step is merely its omission.

If a step abnormally terminates, the system normally bypasses all following st ps in the job. To make the system execute a following step, for instance, to write a dump, code EVEN or ONLY on that step's EXEC statement. The EVEN or ONLY subparameters are interpreted first. If they indicate that the step should be executed, then the return code tests, if specified, are performed. If no return code tests were coded or if none of the coded tests is satisfied, the system executes the step.

Instead of coding a JOB statement COND parameter, code an EXEC statement COND parameter when you want to:

- Specify different tests for each job step.
- Name a specific step whose return code the system is to test.
- Specify special conditions for executing a job step.
- Bypass only one step. When a step is bypassed because of a JOB COND parameter, all following steps in the job are bypassed.

A COND parameter on the first EXEC statement in a job is meaningless.

The tests are made against return codes from the current execution of the job. If a return code test specifies a previous step that was bypassed, the test is ignored.

Note: In both JES2 and JES3 systems, an EXEC COND parameter determines if a step is executed or bypassed. However, JES3 processes all jobs as though each step will execute; therefore, JES3 allocates devices for steps that are bypassed.

EXEC: COND

Syntax:

```
COND[.procstepname]=(code,operator)

COND[.procstepname]=((code,operator[,stepname][.procstepname])
                    [,(code,operator[,stepname][.procstepname])]...
                    [,EVEN]
                    [,ONLY])

COND=EVEN
COND=ONLY
```

- One return code test is: (code,operator)
- You can omit the outer parentheses if you code only one return code test or only EVEN or ONLY.
- Specify up to eight return code tests. However, if you code EVEN or ONLY, specify up to seven return code tests.
- You can omit all return code tests and code only EVEN or ONLY.
- Place the EVEN or ONLY subparameters before, between, or after the return code tests.

Subparameter Definition

code

Specifies a number that the system compares to the return codes from all previous steps in the job or from specific steps. code is a decimal number from 0 through 4095.

Note: Specifying a decimal number greater than 4095 could result in invalid return code testing or invalid return codes in messages.

operator

Specifies the type of comparison to be made to the return code. If the specified test is true, the step is bypassed. Use the chart on page 14-13 to select the correct operator. Operators and their meanings are:

Operator	Meaning
GT	Greater than
GE	Greater than or equal to
EQ	Equal to
NE	Not equal to
LT	Less than
LE	Less than or equal to

stepname

Identifies the EXEC statement of a previous job step that issues the return code to be used in the test. If the specified step is in a procedure, this step must be in the same procedure; otherwise, the specified step must not be in a procedure.

If you omit stepname, the code you specify is compared to the return codes from all previous steps.

stepname.procstepname
Identifies a step in a cataloged or in-stream procedure called by an earlier job step. Stepname identifies the EXEC statement of the calling job step; procstepname identifies the EXEC statement of the procedure step that issues the return code to be used in the test.

EVEN
Specifies that this job step is to be executed **even if** a preceding job step abnormally terminated. When EVEN is coded, the system:

- Does not test the return code of any steps that terminated abnormally.
- Does test the return code of any steps that terminated normally. If none of the return code tests for these steps is satisfied, this job step is executed.

If the operator terminated a job step with a CANCEL command, the system ignores EVEN.

ONLY
Specifies that this job step is to be executed **only if** a preceding step abnormally terminated. When ONLY is coded, the system:

- Does not test the return code of any steps that terminated abnormally.
- Does test the return code of any steps that terminated normally. If none of the return code tests for these steps is satisfied, this job step is executed.

If the operator terminated a job step with a CANCEL command, the system ignores ONLY.

Overrides

If you code the COND parameter on the JOB statement and on one or more of the job's EXEC statements, and if a return code test on the JOB statement is satisfied, the job terminates. In this case, the system ignores any EXEC statement COND parameters.

If the tests on the JOB statement are not satisfied, the system then performs the return code tests on the EXEC statement. If a return code test is satisfied, the step is bypassed.

On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the COND parameter overrides the COND parameter on or is added to:

- The EXEC statement named in the procstepname qualifier, which is to the left of the equals sign. The parameter applies only to the named procedure step. The EXEC statement can have as many COND.procstepname parameters as the procedure has steps; each COND parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to this job step and to all steps in the called procedure.

Cautions when Specifying COND Parameters

Backward References to Data Sets: If a step is bypassed because of its COND parameter or if a step abnormally terminates, a data set that was to have been created or cataloged in the step may not exist, may not be cataloged, or may be incomplete. Thus, a job step should not refer to a data set being created or cataloged in a step that could be bypassed or abnormally terminated.

JOBLIB and COND=ONLY: If the job contains a JOBLIB DD statement and ONLY is specified in a job step, the JOBLIB unit and volume information are not passed to the next step; when the next step is executed, the system searches the catalog for the JOBLIB data set.

Job Time Out: The system abnormally terminates a job with a system completion code of 322 if the EXEC or JOB statement TIME parameter or the default time limit specified at JES initialization is exceeded. This time out occurs regardless of any COND parameters.

When the JOB Statement Contains a RESTART Parameter: When restarting a job, do not specify in the deferred restart step or in any following steps a COND parameter that refers to a stepname or stepname.procstepname for a step before the restart step. The system ignores any COND parameters that refer to preceding steps.

Summary of COND Parameters

Test in COND Parameter	Return Code (RC) from a Previous Step	
	Execute Current Step	Bypass Current Step
COND=(code,GT)	RC ≥ code	RC < code
COND=(code,GE)	RC > code	RC ≤ code
COND=(code,EQ)	RC ¬= code	RC = code
COND=(code,LT)	RC ≤ code	RC > code
COND=(code,LE)	RC < code	RC ≥ code
COND=(code,NE)	RC = code	RC ¬= code

Figure 14-1. Execution or Bypassing of Current Step Based on COND Parameter

EVEN or ONLY Specified?	Any Preceding Abend?	Any Tests Satisfied	Current Step Execute?
EVEN	No	No	Yes
EVEN	No	Yes	No
EVEN	Yes	No	Yes
EVEN	Yes	Yes	No
ONLY	No	No	No
ONLY	No	Yes	No
ONLY	Yes	No	Yes
ONLY	Yes	Yes	No
Neither	No	No	Yes
Neither	No	Yes	No
Neither	Yes	No	No
Neither	Yes	Yes	No

Figure 14-2. Effect of EVEN and ONLY Subparameters on Step Execution

Examples of the COND Parameter

```
//STEP6  EXEC PGM=DISKUTIL,COND=(4,GT,STEP3)
```

In this example, if the return code from STEP3 is 0 through 3, the system bypasses STEP6. If the return code is 4 or greater, the system executes STEP6. Because neither EVEN nor ONLY is specified, the system does not execute this step if a preceding step abnormally terminates.

```
//TEST2  EXEC PGM=DUMPINT,COND=((16,GE),(90,LE,STEP1),ONLY)
```

The system executes this step ONLY if two conditions are met:

1. A preceding job step abnormally terminated.
2. No return code tests are satisfied.

Therefore, the system executes this step only when all three of the following are true:

- A preceding job step abnormally terminated.
- The return codes from all preceding steps are 17 or greater.
- The return code from STEP1 is 89 or less.

The system bypasses this step if any one of the following is true:

- All preceding job steps terminated normally.
- The return code from any preceding step is 0 through 16.
- The return code from STEP1 is 90 or greater.

```
//STEP1  EXEC  PGM=CINDY
          .
          .
//STEP2  EXEC  PGM=NEXT,COND=(4,EQ,STEP1)
          .
          .
//STEP3  EXEC  PGM=LAST,COND=((8,LT,STEP1),(8,GT,STEP2))
          .
```

In this example, if STEP1 returns a code of 4, STEP2 is bypassed. Before STEP3 is executed, the system performs the first return code test. If 8 is less than the return code from STEP1, STEP3 is bypassed; or, restated, if the STEP1 return code is less than or equal to 8, STEP3 is executed. Because 4 is less than 8, STEP3 is executed.

The system does not perform the second return code test because STEP2 was bypassed.

```
//STP4  EXEC PROC=BILLING,COND.PAID=((20,LT),EVEN),
//      COND.LATE=(60,GT,FIND),
//      COND.BILL=((20,GE),(30,LT,CHGE))
```

This statement calls cataloged or in-stream procedure BILLING. The statement specifies different return code tests for each of the procedure steps: PAID, LATE, and BILL. The system executes step PAID even if a preceding step abnormally terminates unless the accompanying return code is satisfied.

DPRTY Parameter

Parameter Type: Keyword, optional

Purpose: Use the DPRTY parameter to assign a dispatching priority to the address space for this job step. The system uses the dispatching priority to determine the order in which to execute tasks.

Syntax:

```
DPRTY[.procstepname]=([value1][,value2])
```

- You can omit the parentheses if you code only value1.
- You must include the parentheses and code a comma before value2 if you code only value2.

Subparameter Definition

value1

Indicates whether this job step is to have the same or a different priority than the job. value1 is a number from 0 through 15.

JES2 determines the job priority from the following, in override order:

1. A JES2 /*PRIORITY statement.
2. A PRTY parameter on the JOB statement.
3. A value calculated from the accounting information on a JES2 /*JOBPARM statement or the JOB statement.
4. An installation default specified at JES2 initialization.

JES3 determines the job priority from the following, in override order:

1. A PRTY parameter on the JOB statement. If the specified priority is invalid, JES3 issues an error message.
2. An installation default specified at JES3 initialization.

value2

Specifies a number to be added to value1 to form the dispatching priority. value2 is a number from 0 through 15. The system forms the internal dispatching priority as follows:

```
dispatching priority = (value1 * 16) + value2
```

EXEC: DPRTY

Defaults

If you omit the DPRTY parameter, the system assigns the job step the automatic priority group (APG) priority.

If you omit value1 or it is equal to the APG priority, the system assigns the step the APG priority and ignores value2. In this case, the system obtains value2 from the installation performance specification (IPS) using the performance group associated with the job step. See *SPL: Initialization and Tuning* for information on IPS. If value2 is not specified in the IPS, the system makes value2 equal to 6.

Relationship to Other Control Statements

For the step, the DPRTY parameter overrides the job's priority.

On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the DPRTY parameter overrides the DPRTY parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The parameter applies only to the named procedure step. The EXEC statement can have as many DPRTY.procstepname parameters as the procedure has steps; each DPRTY parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to this job step and to all steps in the called procedure.

Examples of the DPRTY Parameter

```
//BP2  EXEC PGM=FOUR,DPRTY=(13,9)
```

The system uses the values in this DPRTY parameter to form a dispatching priority for this step. Because the numbers are relatively high, the dispatching priority will be high: 217.

```
//STEP  EXEC  PROC=CLEAN,DPRTY=(11,7)
```

This EXEC statement calls a cataloged procedure named CLEAN, which has three steps. The DPRTY parameter applies to all three steps. The dispatching priority is 183.

```
//STEP  EXEC  PROC=CLEAN,DPRTY.UP=(10,7)
```

In this statement, the DPRTY parameter applies only to the procedure step UP. The dispatching priority for UP is 167.

DYNAMNBR Parameter

Parameter Type: Keyword, optional

Purpose: Use the DYNAMNBR parameter to tell the system to hold a number of resources in anticipation of reuse. Code DYNAMNBR instead of several DD statements with DYNAM parameters.

Syntax:

```
DYNAMNBR[.procstepname]=n
```

Subparameter Definition

n

Specifies a value used to calculate the maximum number of data set allocations that the system can hold in anticipation of reuse. Specify n as a decimal number from 0 through 3273 minus the number of DD statements in the step.

Note that the limit of 3273 is based on the number of single unit DD statements for a 64K TIOT (task input output table). This limit can be different depending on the installation-defined TIOT size. 32K is the default TIOT size. The limit for a 32K TIOT is 1635.

The number of resources that the system actually holds in anticipation of reuse equals n plus the number of DD statements in the step, including any DD statements in a cataloged or in-stream procedure called by the step.

Defaults

If no DYNAMNBR parameter is specified, the default is 0. If you code DYNAMNBR incorrectly, the system uses the default of 0 and issues a JCL warning message.

On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the DYNAMNBR parameter overrides the DYNAMNBR parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The parameter applies only to the named procedure step. The EXEC statement can have as many DYNAMNBR.procstepname parameters as the procedure has steps; each DYNAMNBR parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to all steps in the called procedure.

EXEC: DYNAMNBR

Example of the DYNAMNBR Parameter

```
//STEP1  EXEC PROC=ACCT,DYNAMNBR.CALC=12
```

For the procedure step CALC, this statement specifies that the system should hold the following data set allocations for reuse: 12 plus the number of DD statements following this EXEC statement and the number of DD statements in procedure ACCT.

PARM Parameter

Use the PARM parameter to pass variable information to the processing program executed by this job step. To use the information, the processing program must contain instructions to retrieve the information.

References.: For details on the format of the passed information and its retrieval, see *Supervisor Services and Macro Instructions.*

Syntax:

```
PARM[.procstepname]=information
```

Length: The entire information passed must not exceed 100 characters:

- Including any commas, which are passed to the processing program.
- Excluding any enclosing parentheses or apostrophes, which are not passed.

For example, PARM='P1,123,MT5' is received by the program as P1,123,MT5.

Commas: When the information consists of more than one expression, separate the expressions by commas and enclose the information in parentheses or apostrophes. For example, PARM=(P1,123,MT5) or PARM='P1,123,MT5'.

Special Characters and Blanks: When an expression contains special characters or blanks, enclose it in apostrophes and the information in parentheses or all the information in apostrophes. For example, PARM=(P50,'12+80') or PARM='P50,12+80'.

Code each apostrophe and ampersand that is part of the information as two consecutive apostrophes or ampersands. For example, code 3462&5 as PARM='3462&&5'.

However, if an expression contains a symbolic parameter, code a single ampersand and do **not** enclose the expression in apostrophes, for example, PARM=&UNIT.

Continuation onto Another Statement: Enclose the information in parentheses. End each statement with a comma after a complete expression. For example:

```
//STEP1  EXEC    PGM=WORK,PARM=(DECK,LIST,'LINECNT=80',
//    '12+80',NOMAP)
```

Subparameter Definition

information
Consists of the information to be passed to the processing program.

EXEC: PARM

On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the PARM parameter overrides the PARM parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The information applies only to the named procedure step. The EXEC statement can have as many PARM.procstepname parameters as the procedure has steps; each PARM parameter must specify a unique procstepname.
- The **first** EXEC statement in the procedure if procstepname is not coded; the system **nullifies any PARM parameters on any following EXEC statements in the procedure**. The information applies to only the first step in the called procedure.

Examples of the PARM Parameter

```
//RUN3  EXEC PGM=APG22,PARM='P1,123,P2=5'
```

The system passes P1,123,P2=5 to the processing program named APG22.

```
//    EXEC PROC=PROC81,PARM=MT5
```

The system passes MT5 to the first step of the procedure named PROC81. If PROC81 contains more steps and their EXEC statements contain PARM parameters, the system nullifies those PARM parameters.

```
//STP6  EXEC PROC=ASMFCLG,PARM.LKED=(MAP,LET)
```

The system passes MAP,LET to the procedure step named LKED in procedure ASMFCLG. If any other procedure steps in ASMFCLG contain a PARM parameter, those PARM parameters remain in effect.

```
//RUN4  EXEC PGM=IFOX00,PARM=(NOOBJECT,'LINECNT=50',
//    DECK)
```

The system passes NOOBJECT,LINECNT=50,DECK to processing program IFOX00. Because the PARM parameter is continued on a second statement, the information is enclosed in parentheses; notice that the continuation occurs after a comma following a complete expression.

PERFORM Parameter

Parameter Type: Keyword, optional

Purpose: Use the PERFORM parameter to specify the performance group for the job step. The installation-defined performance groups determine the rate at which associated steps have access to the processor, storage, and channels.

Syntax:

```
PERFORM[.procstepname]=n
```

Subparameter Definition

n

Requests a performance group. The n is a number from 1 through 999 and must identify a performance group that has been defined by your installation. The specified performance group should be appropriate for your step type according to your installation's rules.

Defaults

If no PERFORM parameter is specified or if the specified PERFORM number fails validity checks, the system uses an installation default specified at initialization. If the installation did not specify a default, the system uses a built-in default:

Default	Use
1	For non-TSO job steps
2	For TSO sessions

See *SPL: Initialization and Tuning* for details.

Overrides

A JOB statement PERFORM parameter applies to all steps of the job and overrides any EXEC statement PERFORM parameters.

Code EXEC statement PERFORM parameters when each job step is to execute in a different performance group. The system uses an EXEC PERFORM parameter only when no PERFORM parameter is on the JOB statement and only during the job step.

EXEC: PERFORM

On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the PERFORM parameter overrides the PERFORM parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The parameter applies only to the named procedure step. The EXEC statement can have as many PERFORM.procstepname parameters as the procedure has steps; each PERFORM parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to all steps in the called procedure.

Example of the PERFORM Parameter

```
//STEPA  EXEC PGM=ADDER,PERFORM=60
```

This job step will be run in performance group 60 if it passes validity checks. The installation must have defined the significance of this performance group.

PGM Parameter

Parameter Type: Positional, optional

Purpose: Use the PGM parameter to name the program that the system is to execute. The specified program must be a member of a partitioned data set used as a system library, a private library, or a temporary library.

Syntax:

```
      ┌program-name                          ┐
PGM=  │*.stepname.ddname                     │
      │*.stepname.procstepname.ddname        │
      │JCLTEST                               │
      └JSTTEST                               ┘
```

The EXEC statement parameter field must begin with a PGM parameter or a PROC parameter. These two parameters must not appear on the same EXEC statement.

Subparameter Definition

program-name

Specifies the member name or alias of the program to be executed. The program-name is 1 through 8 alphanumeric or national ($, #, @) characters; the first character must be alphabetic or national ($, #, @).

Use this form of the parameter when the program resides in a system library, such as SYS1.LINKLIB, or in a private library specified in the job by a JOBLIB DD statement or in the step by a STEPLIB DD statement.

***.stepname.ddname**

Refers to a DD statement that defines, as a member of a partitioned data set, the program to be executed. Stepname identifies the EXEC statement of the earlier job step that contains the DD statement with ddname in its name field.

Use this form of the parameter when a previous job step creates a temporary library to store a program until it is required.

When referring to a DD statement, the system does not honor requests for special program properties as defined in the program properties table (PPT). (See *SPL: System Modifications.*)

***.stepname.procstepname.ddname**

Refers to a DD statement that defines, as a member of a partitioned data set, the program to be executed. The DD statement is in a cataloged or in-stream procedure that is called by an earlier job step. Stepname identifies the EXEC statement of the calling job step; procstepname identifies the EXEC statement of the procedure step that contains the DD statement with ddname in its name field.

Use this form of the parameter when a previous job step calls a procedure that creates a temporary library to store a program until it is required.

When referring to a DD statement, the system does not honor requests for special program properties as defined in the program properties table (PPT). (See *SPL: System Modifications.*)

JCLTEST (JES3 only)
JSTTEST (JES3 only)
Requests that the system scan the step's job control statements for syntax errors without executing the job or allocating devices. JCLTEST or JSTTEST provides for a step the same function as provided by the JOB statement TYPRUN=SCAN parameter for a job. See *JES3 Diagnosis* for details.

Note: JCLTEST and JSTTEST are supported only in JES3 systems.

Examples of the PGM Parameter

```
//JOB8    JOB  ,BOB,MSGLEVEL=(2,0)
//JOBLIB  DD   DSNAME=DEPT12.LIB4,DISP=(OLD,PASS)
//STEP1   EXEC PGM=USCAN
```

These statements indicate that the system is to search the private library DEPT12.LIB4 for the member named USCAN, read the member into storage, and execute the member.

```
//PROCESS JOB  ,MARY,MSGCLASS=A
//CREATE  EXEC PGM=IEWL
//SYSLMOD DD   DSNAME=&&PARTDS(PROG),UNIT=3350,DISP=(MOD,PASS),
//             SPACE=(1024,(50,20,1))
//GO      EXEC PGM=*.CREATE.SYSLMOD
```

The EXEC statement named GO contains a backward reference to DD statement SYSLMOD, which defines a library created in the step named CREATE. Program PROG is a member of the partitioned data set &&PARTDS, which is a temporary data set. Step GO executes program PROG. The data set &&PARTDS is deleted at the end of the job.

```
//JOBC    JOB  ,JOHN,MSGCLASS=H
//STEP2   EXEC PGM=UPDT
//DDA     DD   DSNAME=SYS1.LINKLIB(P40),DISP=OLD
//STEP3   EXEC PGM=*.STEP2.DDA
```

The EXEC statement named STEP3 contains a backward reference to DD statement DDA, which defines system library SYS1.LINKLIB. Program P40 is a member of SYS1.LINKLIB; STEP3 executes program P40.

PROC and Procedure Name Parameters

Parameter Type: Positional, optional

Purpose: Use the PROC parameter to specify that the system is to call and execute a cataloged or in-stream procedure.

Syntax:

```
{PROC=procedure-name}
{procedure-name     }
```

- The EXEC statement parameter field must begin with a PGM parameter or a PROC parameter. These two parameters must not appear on the same EXEC statement.
- You can omit PROC= and code only the procedure-name.

Subparameter Definition

procedure-name
Identifies the procedure to be called and executed:

- The member name or alias of a cataloged procedure.
- The name on the PROC statement that begins an in-stream procedure. The in-stream procedure must appear earlier in this job.

The procedure-name is 1 through 8 alphanumeric or national ($, #, @) characters; the first character must be alphabetic or national ($, #, @).

Effect of PROC Parameter on Other Parameters and Following Statements

Because this EXEC statement calls a cataloged or in-stream procedure, the other parameters on the statement are added to or override corresponding parameters on the EXEC statements in the called procedure. See the descriptions of the other parameters for details of their effects.

Any DD statements following this EXEC statement are added to the procedure, or override or nullify corresponding DD statements in the procedure.

Examples of the PROC Parameter

```
//SP3  EXEC PROC=PAYWKRS
```

This statement calls the cataloged or in-stream procedure named PAYWKRS.

```
//BK  EXEC OPERATE
```

This statement calls the cataloged or in-stream procedure named OPERATE.

RD Parameter

Parameter Type: Keyword, optional

Purpose: Use the RD (restart definition) parameter to:

- Specify that the operator is to perform automatic step restart if the job fails.
- Suppress, partially or totally, the action of the assembler language CHKPT macro instruction or the DD statement CHKPT parameter.

The system can perform automatic restart only if all of the following are true:

- The JOB or EXEC statement contains RD=R or RD=RNC.
- The step to be restarted returned a completion code that indicated no error in the step.
- The operator authorizes a restart.
- The job has a job journal.

A job journal is a sequential data set that contains job-related control blocks needed for restart.

For JES2, specify a job journal by one of the following:

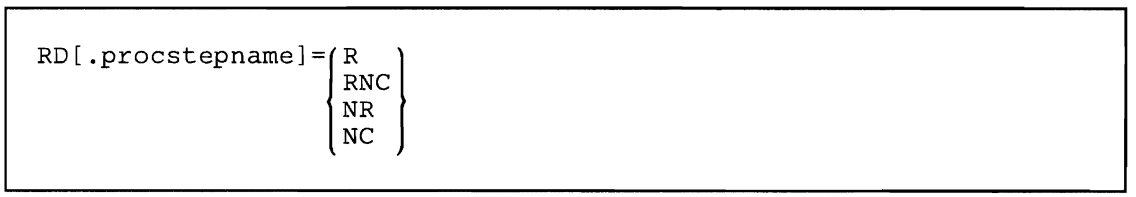
- An installation option during JES2 initialization.
- RD=R or RD=RNC on either the JOB statement or any one EXEC statement in the job.

For JES3, specify a job journal in one of the following:

- An installation option during JES3 initialization.
- RD=R or RD=RNC on either the JOB statement or any one EXEC statement in the job.
- JOURNAL=YES on a JES3 //*MAIN statement in the job.

References: For detailed information on deferred checkpoint restart, see *Checkpoint/Restart User's Guide*.

Syntax:



Subparameter Definition

R (Restart, Checkpoints Allowed)
Indicates that the operator is to perform automatic step restart if the job step fails.

RD=R does not suppress checkpoint restarts:

- If the processing program executed in a job step does not include a CHKPT macro instruction, RD=R allows the system to restart execution at the beginning of the abnormally terminated step.
- If the program includes a CHKPT macro instruction, RD=R allows the system to restart execution at the beginning of the step, if the step abnormally terminates before the CHKPT macro instruction is executed.
- If the step abnormally terminates after the CHKPT macro instruction is executed, only checkpoint restart can occur. If you cancel the affects of the CHKPT macro instruction before the system performs a checkpoint restart, the request for automatic step restart is again in effect.

RNC (Restart, No Checkpoints)
Indicates that the operator is to perform automatic step restart if the job step fails.

RD=RNC suppresses automatic and deferred checkpoint restarts. It suppresses:

- Any CHKPT macro instruction in the processing program: That is, the operator is not to perform an automatic checkpoint restart, and the system is not to perform a deferred checkpoint restart if the job is resubmitted.
- The DD statement CHKPT parameter.
- The checkpoint at end-of-volume (EOV) facility.

NR (No Automatic Restart, Checkpoints Allowed)
Indicates that the operator is **not** to perform automatic step restart if the job fails.

RD=NR suppresses automatic checkpoint restart but permits deferred checkpoint restarts. It permits:

- A CHKPT macro instruction to establish a checkpoint.
- The job to be resubmitted for restart at the checkpoint. On the JOB statement when resubmitting the job, specify the checkpoint in the RESTART parameter.

If you code RD=NR and the system fails, RD=NR does not prevent the job from restarting.

EXEC: RD

NC (No Automatic Restart, No Checkpoints)
Indicates that the operator is **not** to perform automatic step restart if the job step fails.

RD=NC suppresses automatic and deferred checkpoint restarts. It suppresses:

- Any CHKPT macro instruction in the processing program.
- The DD statement CHKPT parameter.
- The checkpoint at EOV facility.

Defaults

If you do not code the RD parameter, the system uses the installation default from the job's job class specified at initialization.

Overrides

- A JOB statement RD parameter applies to all steps of the job and overrides any EXEC statement RD parameters.

  When no RD parameter is on the JOB statement, the system uses an EXEC statement RD parameter, but only during the job step. Code EXEC statement RD parameters when you want to specify different restart types for each job step.

- A request by a CHKPT macro instruction for an automatic checkpoint restart overrides a request by a JOB or EXEC statement RD=R parameter for automatic step restart.

Relationship to Other Control Statements

Code RD=NC or RD=RNC to suppress the action of the DD statement CHKPT parameter.

On EXEC Statement that Calls a Procedure

If the EXEC statement calls a cataloged or in-stream procedure, the RD parameter is added to or overrides the RD parameter on:

- The EXEC statement named in the procstepname qualifier. The information applies only to the named procedure step. The EXEC statement can have as many RD.procstepname parameters as the procedure has steps; each RD parameter must specify a unique procstepname.

- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to all steps in the called procedure.

Examples of the RD Parameter

```
//STEP1  EXEC PGM=GIIM,RD=R
```

RD=R specifies that the operator is to perform automatic step restart if the job step fails.

```
//NEST  EXEC PGM=T18,RD=RNC
```

RD=RNC specifies that, if the step fails, the operator is to perform automatic step restart. RD=RNC suppresses automatic and deferred checkpoint restarts.

```
//CARD  EXEC PGM=WTE,RD=NR
```

RD=NR specifies that the operator is not to perform automatic step restart or automatic checkpoint restart. However, a CHKPT macro instruction can establish checkpoints to be used later for a deferred restart.

```
//STP4  EXEC PROC=BILLING,RD.PAID=NC,RD.BILL=NR
```

This statement calls a cataloged or in-stream procedure BILLING. The statement specifies different restart requests for each of the procedure steps: PAID and BILL.

REGION Parameter

Parameter Type: Keyword, optional

Purpose: Use the REGION parameter to specify the amount of space that the step requires.

The specified or default region size sets an upper boundary to limit region size for variable-length GETMAINs. The system uses the upper boundary for variable-length GETMAINs as long as the region still has available at least the minimum amount of storage requested.

In addition, the IBM- or installation-supplied routine IEFUSI uses the region size to establish a second limiting value. 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 variable-length GETMAINs or the amount requested for a fixed-length GETMAIN exceeds this second value, the job step abnormally terminates.

Specifying a Region Size Value: If your installation does not change the IBM-supplied default limits in the IEALIMIT and/or IEFUSI exit routine modules, then specifying various values for the region size have the following results:

- A value equal to 0K or 0M -- gives the job step all the storage available below and above 16 megabytes. The resulting size of the region below and above 16 megabytes is unpredictable.
- A value greater than 0K or 0M and less than or equal to 16384K or 16M -- establishes the size of the private area below 16 megabytes. If the region size specified is not available below 16 megabytes, the job step abnormally terminates. The extended region size is the default value of 32 megabytes.
- A value greater than 16384K or 16M and less than or equal to 32768K or 32M -- gives the job step all the storage available below 16 megabytes. The resulting size of the region below 16 megabytes is unpredictable. The extended region size is the default value of 32 megabytes.
- A value greater than 32768K or 32M and less than or equal to 2096128K or 2047M -- gives the job step all the storage available below 16 megabytes. The resulting size of the region below 16 megabytes is unpredictable. The extended region size is the specified value. If the region size specified is not available above 16 megabytes, the job step abnormally terminates.

References: For more information on the region size, see *SPL: System Modifications.* For more information on region size with checkpoint/restart jobs, see *Checkpoint/Restart User's Guide.*

Syntax:

```
REGION[.procstepname]=(valueK)
                      (valueM)
```

Subparameter Definition

valueK
Specifies the required storage in kilobytes (1 kilobyte = 1024 bytes). The value is 1 through 7 decimal numbers, from 1 through 2096128. Code an even number. For example, REGION=66K. If you code an odd number, the system treats it as the next highest even number.

valueM
Specifies the required storage in megabytes (1 megabyte = 1024 kilobytes). The value is 1 through 4 decimal numbers, from 1 through 2047. Code either an even or odd number. For example, REGION=3M.

Defaults

If no REGION parameter is specified, the system uses an installation default specified at JES initialization.

Overrides

A JOB statement REGION parameter applies to all steps of the job and overrides any EXEC statement REGION parameters.

When no REGION parameter is on the JOB statement, the system uses an EXEC statement REGION parameter, but only during the job step. Code EXEC statement REGION parameters when you want to specify a different region size for each job step.

Relationship to the EXEC ADDRSPC Parameter

When ADDRSPC=REAL: Code a REGION parameter to specify how much real storage the job needs. If you omit the REGION parameter, the system uses the default.

When ADDRSPC=VIRT or ADDRSPC is Omitted: Code a REGION parameter to specify how much virtual storage the job needs. If you omit the REGION parameter, the system uses the default.

On EXEC Statement that Calls a Procedure

If the EXEC statement calls a cataloged or in-stream procedure, the REGION parameter is added to or overrides the REGION parameter on:

- The EXEC statement named in the procstepname qualifier. The information applies only to the named procedure step. The EXEC statement can have as many REGION.procstepname parameters as the procedure has steps; each REGION parameter must specify a unique procstepname.

- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to all steps in the called procedure.

Examples of the REGION Parameter

```
//MKBOYLE EXEC PROC=A,ADDRSPC=REAL,REGION=40K
```

The system assigns 40K bytes of real storage to this job step.

```
//STP6  EXEC PGM=CONT,REGION=120K
```

The system assigns a region of 120K bytes. When the ADDRSPC parameter is not specified, the system defaults to ADDRSPC=VIRT.

TIME Parameter

Parameter Type: Keyword, optional

Purpose: Use the TIME parameter to specify the maximum length of time that a job step is to use the processor and to find out through messages how much processor time the step used.

A step that exceeds its allotted time abnormally terminates and causes termination of the job, unless a user exit routine extends the time for the job. The exit routine is established through System Management Facilities (SMF).

References: See *SPL: System Management Facilities (SMF).*

Syntax:

```
TIME[.procstepname]={([minutes][,seconds])}
                    {1440                 }
```

You can omit the parentheses if you code only 1440 or the processor time in minutes.

Subparameter Definition

minutes
Specifies the maximum number of minutes the step can use the processor. The minutes must be a number from 1 through 1439.

Code TIME=0 on the EXEC statement to indicate that the step is to use the unexpired time from the previous step. If this step exceeds that unexpired time, it abnormally terminates.

seconds
Specifies the maximum number of seconds that the step can use the processor, in addition to any minutes that are specified. The seconds must be a number from 1 through 59.

1440
Indicates that the step can use the processor for an unlimited amount of time; 1440 literally means 24 hours. Code TIME=1440 for the following reasons:

- To obtain job step accounting information.
- To specify that the system is to allow this step to remain in a continuous wait state for more than the installation time limit, which is established through SMF.

EXEC: TIME

Defaults

If no TIME parameter is specified, JES uses an installation default specified at initialization.

If 1440 is not specified, SMF uses its current job wait time limit.

Overrides

For a JOB statement TIME parameter other than TIME=1440, the system sets the time limit for each step to:

- The step time limit specified on the EXEC statement TIME parameter or the job time remaining after execution of previous job steps, whichever is smaller.
- If no EXEC TIME parameter was specified, (1) the default time limit or (2) the job time remaining after execution of previous steps, whichever is smaller.

On EXEC Statement that Calls a Procedure

If the EXEC statement calls a cataloged or in-stream procedure, the TIME parameter is added to or overrides the TIME parameter on:

- The EXEC statement named in the procstepname qualifier. The information applies only to the named procedure step. The EXEC statement can have as many TIME.procstepname parameters as the procedure has steps; each TIME parameter must specify a unique procstepname.

If procstepname is not coded, the TIME parameter applies to the entire procedure and nullifies any TIME parameters on EXEC statements in the procedure.

Time Handling

How the System Converts the Time Value: The job time limit or the time remaining after execution of previous steps in a job is converted by the system to seconds and then rounded to the nearest unit, where 1 unit = 1.048576 seconds. Thus a step can begin execution with up to one-half unit more or one-half unit less time than expected. If the time remaining for the job is less than one-half unit, a step will begin execution with zero time, resulting in an abnormal termination.

Time Checking: Because the system checks the processor time-used field about every 10.5 seconds, the actual time that a job uses the processor can exceed the specified TIME value by up to 10.5 seconds. For example, the system checks the job's time-used field and finds 0.5 seconds remaining. Because the system does not again check the job's time-used field for about 10.5 seconds, the job can execute for an additional 10.5 seconds and thus exceed the coded TIME value by 10 seconds.

Examples of the TIME Parameter

```
//STEP1  EXEC PGM=GRYS,TIME=(12,10)
```

This statement specifies that the maximum amount of time the step can use the processor is 12 minutes, 10 seconds.

```
//FOUR  EXEC PGM=JPLUS,TIME=(,30)
```

This statement specifies that the maximum amount of time the step can use the processor is 30 seconds.

```
//INT  EXEC PGM=CALC,TIME=5
```

This statement specifies that the maximum amount of time the step can use the processor is 5 minutes.

```
//LONG  EXEC PGM=INVANL,TIME=1440
```

This statement specifies that the step can have unlimited use of the processor. Therefore, the step can use the processor and can remain in a wait state for an unspecified period of time, if not restricted by the JOB statement TIME parameter.

```
//STP4  EXEC PROC=BILLING,TIME.PAID=(45,30),TIME.BILL=(112,59)
```

This statement calls cataloged or in-stream procedure BILLING. The statement specifies different time limits for each of the procedure steps: PAID and BILL.

For examples of TIME coded on both the JOB and EXEC statements, see "Examples of the TIME Parameter on JOB and EXEC Statements" on page 15-40.

Chapter 15. JOB Statement

Purpose: Use the JOB statement to mark the beginning of a job and to tell the system how to process the job. Also, when jobs are stacked in the input stream, the JOB statement marks the end of the preceding job.

The parameters you can specify for job processing are arranged alphabetically in the following pages.

References: For information about the JES initialization parameters that provide installation defaults, see *SPL: JES2 Initialization and Tuning* and *SPL: JES3 Initialization and Tuning*.

Syntax:

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

The JOB statement consists of the characters // in columns 1 and 2 and four fields: name, operation (JOB), parameter, and comments. Do not code comments if the parameter field is blank.

A JOB statement is required for each job.

Name Field

Code a jobname on every JOB statement, as follows:

- Each jobname must be unique.
- The jobname must begin in column 3.
- The jobname is 1 through 8 alphanumeric or national ($, #, @) characters. If your system uses ANSI tapes, the jobname must contain only alphanumeric characters; it must not contain national ($, #, @) characters.
- The first character must be alphabetic or national ($, #, @).
- The jobname must be followed by at least one blank.

JOB

Operation Field

The operation field consists of the characters JOB and must be preceded and followed by at least one blank. It can begin in any column.

Parameter Field

A JOB statement has two kinds of parameters: positional and keyword. All parameters are optional; however, your installation may require the accounting information parameter and the programmer's name parameter.

Positional Parameters: A JOB statement can contain two positional parameters. They must precede all keyword parameters. You must code the accounting parameter first, followed by the programmer's name parameter.

POSITIONAL PARAMETERS	VALUES	PURPOSE
([account-number][,accounting-information]...) See page 15-5	account-number,accounting-information: up to 142 characters	Specifies an account number and other accounting information, formatted as required by the installation. This parameter may be required by the installation.
programmer's-name See page 15-26	programmer's name: 1 - 20 characters	Identifies the owner of the job. This parameter may be required by the installation.

Keyword Parameters: A JOB statement can contain the following keyword parameters. You can code any of the keyword parameters in any order in the parameter field after the positional parameters.

KEYWORD PARAMETERS	VALUES	PURPOSE
ADDRSPC = {VIRT / REAL} See page 15-9	VIRT: virtual (pageable) storage REAL: real (nonpageable) storage	Indicates the type of storage required for the job.
CLASS = jobclass See page 15-11	jobclass: A - Z, 0 - 9	Assigns the job to a job class.
COND = ((code,operator)[,(code,operator)]...) See page 15-13	code: 0 - 4095 operator: GT, GE, EQ, NE, LT, LE (Code from chart on page 15-14)	Specifies the return code tests used to determine whether a job will continue processing or be terminated.

KEYWORD PARAMETERS	VALUES	PURPOSE
GROUP = group-name See page 15-15	group-name: 1 - 8 alphanumeric or $, #, @ characters	Identifies a group to which a RACF-defined user is to be connected.
MSGCLASS = class See page 15-17	class: A - Z, 0 - 9	Assigns the job log to an output class.
MSGLEVEL = ([statements][,messages]) See page 15-19	statements: 0 Only JOB statement 1 All JCL and procedure statements 2 Only JCL statements messages: 0 Only JCL messages 1 JCL, JES, and operator messages	Indicates the job control information to be printed in the job log.
NOTIFY = userid See page 15-21	userid: 1 - 7 alphanumeric characters	Requests that the system send a message to a TSO userid when this background job completes.
PASSWORD = (password[,new-password]) See page 15-23	password or new-password: 1 - 8 alphanumeric or $, #, @ characters	Identifies the current RACF password or specifies a new RACF password.
PERFORM = n See page 15-25	n: 1 - 999	Specifies the job's performance group, which determines the rate at which the job has access to the processor, storage, and channels.
PRTY = priority See page 15-28	priority (JES2): 0 - 15 priority (JES3): 0 - 14	JES2: Assigns the job's queue selection priority. JES3: Assigns the job's initiation or selection priority in its job class.
RD = { R | RNC | NR | NC } See page 15-30	R: restart, checkpoints allowed RNC: restart, no checkpoints NR: no restart, checkpoints allowed NC: no restart, no checkpoints	Indicates whether the operator should perform automatic step restart, if the job fails, and controls whether checkpoints are written for CHKPT macros or DD statement CHKPT parameters.
REGION = { valueK | valueM } See page 15-33	valueK: even number, 1 - 7 digits from 1 - 2096128 valueM: even or odd number, 1 - 4 digits from 1 - 2047	Specifies the amount of space in kilobytes or megabytes required by the job.

JOB

KEYWORD PARAMETERS	VALUES	PURPOSE
RESTART = ({ * / stepname / stepname.procstepname } [,checkid]) See page 15-35	*: at first step stepname: at named step procstepname: step is in named procedure checkid: at checkpoint in first or named step	Specifies restart of a job at the beginning of a step or from a checkpoint within a step.
TIME = { ([minutes][,seconds]) / 1440 } See page 15-38	minutes: 1 - 1439 seconds: 1 - 59	Specifies the maximum time the job is to use the processor and requests messages giving the time used.
TYPRUN = { COPY / HOLD / JCLHOLD / SCAN } See page 15-41	COPY: copies job stream to sysout data set (JES2 only) HOLD: holds job JCLHOLD: holds job before JCL processing (JES2 only) SCAN: scans JCL for syntax errors	Requests special job processing.
USER = userid See page 15-43	userid: 1 - 7 alphanumeric or $, #, @ characters	Identifies the job's owner to RACF, SRM, and other system components.

Comments Field

The comments field follows the parameter field after at least one intervening blank. If you do not code any parameters on a JOB statement, do not code any comments.

Location in the JCL

A JOB statement must be the first statement in each job. JOB statements never appear in cataloged or in-stream procedures.

Examples of JOB Statements

```
//ALPHA    JOB   843,LINLEE,CLASS=F,MSGCLASS=A,MSGLEVEL=(1,1)
//LOS      JOB   ,'J M BUSKIRK',TIME=(4,30),MSGCLASS=H,MSGLEVEL=(2,0)
//MART     JOB   1863,RESTART=STEP4 THIS IS THE THIRD JOB STATEMENT.
//TRY8     JOB
//RACF1    JOB   'D83,123',USER=RAC01,GROUP=A27,PASSWORD=XYY
//RUN1     JOB   'D8306P,D83,B1062J12,S=C','JUDY PERLMAN',MSGCLASS=R,
//       MSGLEVEL=(1,1),CLASS=3,NOTIFY=D83JCS1,
//       COND=(8,LT)
```

Accounting Information Parameter

Parameter Type: Positional, required (according to installation procedures)

Purpose: Use the accounting information parameter to enter an account number and any other accounting information that your installation requires.

References: For more information on how to add accounting routines, see *SPL: System Management Facilities.*

Syntax:

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

Location: Code the accounting information parameter first in the parameter field.

Omission: If you omit the accounting information parameter but you are coding a programmer's name parameter, code a comma to indicate the omitted parameter. If you omit both positional parameters, do not code any commas before the first keyword parameter.

Length: The entire accounting information parameter must not exceed 142 characters:

- Including any commas, which are considered part of the information.
- Excluding any enclosing parentheses, which are not considered part of the information.

Multiple Subparameters: When the accounting information parameter consists of more than one subparameter, separate the subparameters by commas and enclose the parameter in parentheses or apostrophes. For example, (5438,GROUP6) or '5438,GROUP6'. If you use apostrophes, all information inside the apostrophes is considered one field.

Special Characters: When a subparameter contains special characters, other than hyphens, enclose it in apostrophes and the entire parameter in parentheses or enclose all of the parameter in apostrophes. For example, (12A75,'DEPT/D58',706) or '12A75,DEPT/D58,706'.

Code each apostrophe or ampersand that is part of the accounting information as two consecutive apostrophes or ampersands. For example, code DEPT'D58 as (12A75,'DEPT''D58',706) or '12A75,DEPT''D58,706'. Code 34&251 as '34&&251'.

Continuation onto Another Statement: Enclose the accounting information parameter in parentheses. End each statement with a comma after a complete subparameter. For example:

```
//JOB1  JOB  (12A75,'DEPT/D58',
//       706)
```

JOB: Accounting Information

Subparameter Definition

account-number
Specifies an accounting number, as defined by the installation.

accounting-information
Specifies more information, as defined by the installation. For example, your department and room numbers.

Relationship to Other Control Statements

If you are to provide accounting information for an individual step within a job, code an ACCT parameter on the EXEC statement for that step.

JES2 Accounting Information Format

Except for the first subparameter, the JES2 accounting information shown in the syntax can, alternatively, appear on the JES2 /*JOBPARM statement. If you code the accounting information parameter in the JES2 format, JES2 can interpret and use it.

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

Syntax:

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

Code a comma in place of each omitted subparameter when other subparameters follow.

Subparameter Definition

pano
Specifies the programmer's accounting number. pano is 1 through 4 alphanumeric characters.

room
Specifies the programmer's room number. room is 1 through 4 alphanumeric characters.

time
Specifies the estimated execution time in minutes. time is 1 through 4 decimal numbers. For example, code **30** for 30 minutes. If you omit a time subparameter and a TIME parameter on the JES2 /*JOBPARM statement, JES2 uses an installation default specified at initialization. If job execution exceeds the time, JES2 sends a message to the operator.

lines
Specifies the estimated line count, in thousands of lines, from this job's sysout data sets. lines is 1 through 4 decimal numbers. For example, code **5** for 5000 lines. If you omit lines, JES2 uses an installation default specified at initialization.

cards

Specifies the estimated number of cards JES2 is to punch from this job's sysout data sets. cards is 1 through 4 decimal numbers. If you omit cards, JES2 uses an installation default specified at initialization.

forms

Specifies the forms that JES2 is to use for printing this job's sysout data sets. forms is 1 through 4 alphanumeric characters. For example, code **5** for 5-part forms. If you omit forms, JES2 uses an installation default specified at initialization.

copies

Specifies the number of times JES2 is to print and/or punch this job's sysout data sets. copies is 1 through 3 decimal numbers not exceeding an installation-specified limit. The maximum is 255. For example, code **2** for two copies. If you omit copies, JES2 assumes one copy.

The copies subparameter is ignored and only one copy is produced if the output class for the job log, as specified in the JOB MSGCLASS parameter, or the output class of any of the job's system output data sets is a held class.

log

Specifies whether or not JES2 is to print the job log. Code **N** to request no job log. If you code any other character or omit this subparameter, JES2 prints the job log. If your installation specified NOLOG for this job's class during JES2 initialization, JES2 will not print a job log.

linect

Specifies the number of lines JES2 is to print per page for this job's sysout data sets. linect is 1 through 3 decimal numbers. When you send a data set across a network, linect cannot exceed 254. When you print the data set locally, linect cannot exceed 255. If you omit linect, JES2 uses an installation default specified at initialization. If you code a zero, JES2 does not eject to a new page when the number of lines exceeds the installation default.

Invalid Subparameters: Your installation can initialize JES2 to do one of the following if the accounting information contains subparameters that are invalid to JES2:

- Ignore the invalid subparameters.
- Terminate the job. In this case, JES2 requires the first two subparameters: **pano** and **room**.

Overrides: A parameter on any of the following statements overrides an equivalent accounting information subparameter on the JOB statement:

- JES2 /*JOBPARM statement
- JES2 /*OUTPUT statement
- OUTPUT JCL statement
- DD statement

Examples of the Accounting Information Parameter

```
//JOB43  JOB  D548-8686
```

```
//JOB44  JOB  (D548-8686,'12/8/85',PGMBIN)
```

Because this statement contains an account-number plus additional accounting-information, parentheses are required.

```
//JOB45  JOB  (CFH1,2G14,15,,,,2)
```

This statement shows a JES2 accounting information parameter: programmer's accounting number, CFH1; room number, 2G14; estimated job time, 15 minutes; and copies, 2. Parentheses are required. Standard values are assumed for the other JES2 subparameters.

ADDRSPC Parameter

Parameter Type: Keyword, optional

Purpose: Use the ADDRSPC parameter to indicate to the system that the job requires virtual storage (pageable) or real storage (nonpageable).

Syntax:

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

Subparameter Definition

VIRT
Requests virtual storage. The system **can** page the job.

REAL
Requests real storage. The system **cannot** page the job and must place each step of the job in real storage.

Defaults

If no ADDRSPC parameter is specified, the default is VIRT.

Overrides

The JOB statement ADDRSPC parameter applies to all steps of the job and overrides any EXEC statement ADDRSPC parameters.

Code EXEC statement ADDRSPC parameters when each job step requires different types of storage. The system uses an EXEC statement ADDRSPC parameter only when no ADDRSPC parameter is on the JOB statement and only during the job step.

Relationship to the JOB REGION Parameter

When ADDRSPC=REAL: Code a REGION parameter to specify how much real storage the job needs. If you omit the REGION parameter, the system uses an installation default specified at JES initialization.

When ADDRSPC=VIRT or ADDRSPC is Omitted: Code a REGION parameter to specify how much virtual storage the job needs. If you omit the REGION parameter, the system uses an installation default specified at JES initialization.

Examples of the ADDRSPC Parameter

```
//PEH  JOB  ,BAKER,ADDRSPC=VIRT
```

The ADDRSPC parameter requests virtual (pageable) storage. The space available to the job is the installation-specified default.

```
//DEB  JOB  ,ERIC,ADDRSPC=REAL,REGION=100K
```

The ADDRSPC parameter requests real (nonpageable) storage. The REGION parameter specifies 100K of storage for the job.

CLASS Parameter

Parameter Type: Keyword, optional

Purpose: Use the CLASS parameter to assign the job to a class. The class you should request depends on the characteristics of the job and your installation's rules for assigning classes.

In a JES2 system, the assigned job class can affect whether or how a job is executed. A job class can be defined during JES2 initialization as:

- Held. The system holds any job assigned to this class until the operator releases it.
- To be copied only. The system copies the input stream for the job directly to a sysout data set and schedules the sysout data set for output processing. The system does not execute the job or allocate devices.
- To be scanned for job control statement syntax errors. The system does not execute the job or allocate devices.

Syntax:

```
CLASS=jobclass
```

Subparameter Definition

jobclass
Identifies the class for the job. The jobclass is one character, A through Z or 0 through 9, and must be a valid class specified at JES initialization.

Defaults

If you do not specify a class, JES uses the installation default specified at initialization, as follows:

- In a JES2 system, the default is based on the source of the job: The system makes the job's class the same as the installation-specified default class for the particular card reader, work station, or time-sharing user that submitted the job.
- In a JES3 system, the default is an installation-defined standard default class.

Overrides

A JES3 //*MAIN statement CLASS parameter overrides a JOB statement CLASS parameter.

JOB: CLASS

Relationship to Other Control Statements

In JES3 systems, you can also code a CLASS parameter on a JES3 //*MAIN statement.

Example of the CLASS Parameter

```
//SETUP  JOB  1249,SMITH,CLASS=M
```

This statement assigns the job to class M.

COND Parameter

Parameter Type: Keyword, optional

Purpose: Use the COND parameter to specify the return code tests the system uses to determine whether a job will continue processing. Before and after each job step is executed, the system performs the COND parameter tests against the return codes from completed job steps. If none of these tests is satisfied, the system executes the job step; if any test is satisfied, the system bypasses all remaining job steps and terminates the job.

The tests are made against return codes from the current execution of the job. A step bypassed because of an EXEC statement COND parameter does not produce a return code.

Bypassing a step because of a return code test is not the same as abnormally terminating the step. The system abnormally terminates a step following an error so serious that it prevents successful execution. In contrast, bypassing of a step is merely its omission.

Note: In both JES2 and JES3 systems, a JOB COND parameter determines if steps are executed or bypassed. However, JES3 processes all jobs as though each step will execute; therefore, JES3 allocates devices for steps that are bypassed.

Syntax:

```
COND=(code,operator)
COND=((code,operator)[,(code,operator)]...)
```

- One return code test is: (code,operator)
- You can omit the outer parentheses if you code only one return code test.
- Specify up to eight return code tests for a job.

Subparameter Definition

code

Specifies a number that the system compares to the return code from each job step. code is a decimal number from 0 through 4095.

Note: Specifying a decimal number greater than 4095 could result in invalid return code testing or invalid return codes in messages.

operator

Specifies the type of comparison to be made to the return code. If the specified test is true, the system bypasses all remaining job steps. Use the chart on this page to select the correct operator. Operators and their meanings are:

Operator	Meaning
GT	Greater than
GE	Greater than or equal to
EQ	Equal to
NE	Not equal to
LT	Less than
LE	Less than or equal to

JOB: COND

Overrides

If you code the COND parameter on the JOB statement and on one or more of the job's EXEC statements, and if a return code test on the JOB statement is satisfied, the job terminates. In this case, the system ignores any EXEC statement COND parameters.

If the tests on the JOB statement are not satisfied, the system then performs the return code tests on the EXEC statement. If an EXEC return code test is satisfied, the step is bypassed.

Summary of COND Parameters

Test in COND Parameter	Return Code (RC) from Just Completed Step	
	Continue Job	**Terminate Job**
COND=(code,GT)	**RC ≥ code**	**RC < code**
COND=(code,GE)	**RC > code**	**RC ≤ code**
COND=(code,EQ)	**RC ¬= code**	**RC = code**
COND=(code,LT)	**RC ≤ code**	**RC > code**
COND=(code,LE)	**RC < code**	**RC ≥ code**
COND=(code,NE)	**RC = code**	**RC ¬= code**

Figure 15-1. Continuation or Termination of the Job Based on COND Parameter

Examples of the COND Parameter

```
//TYPE  JOB  (611,402),BOURNE,COND=(7,LT)
```

The COND parameter specifies that if 7 is less than the return code, the system terminates the job. Any return code less than or equal to 7 allows the job to continue.

```
//TEST  JOB  501,BAXTER,COND=((20,GE),(30,LT))
```

The COND parameter specifies that if 20 is greater than or equal to the return code or if 30 is less than the return code, the system terminates the job. Any code of 21 through 30 allows the job to continue.

GROUP Parameter

Parameter Type: Keyword, optional

Purpose: Use the GROUP parameter to specify a RACF-defined group to which a RACF-defined user is to be connected. RACF places each RACF-defined user in a default group; the GROUP parameter is needed only to specify a group other than a user's default group.

If the installation contains the feature for propagation of the user and group identification, the USER, the PASSWORD, and, optionally, the GROUP parameters are required on JOB statements only for the following:

- Batch jobs submitted through an input stream, such as a card reader, (1) if the job requires access to RACF-protected resources or (2) if the installation requires that all jobs have RACF identification.
- Jobs submitted by one TSO user for another user. In this case, the JOB statement must specify the other user's userid and password. The group id is optional.
- Jobs that execute at another network node that uses RACF protection.

Otherwise, the USER, PASSWORD, and GROUP parameters can be omitted from JOB statements. RACF uses the userid, password, and default group id of the submitting TSO user or job.

References: For more information on RACF-protected facilities, see *Resource Access Control Facility (RACF) General Information Manual.*

Syntax:

```
GROUP=group-name
```

Subparameter Definition

group-name
Identifies the group with which the system is to associate the user. group-name is 1 through 8 alphanumeric or national ($, #, @) characters. The first character must be alphabetic or national ($, #, @).

Defaults

If you do not code the GROUP parameter, but do code the USER and PASSWORD parameters, the system assigns the RACF default group name associated with the specified userid. However, the default group name is not passed to JES and thus is not available to JES installation exits.

JOB: GROUP

Relationship to Other Parameters

If the JOB statement contains a GROUP parameter, the statement must also contain USER and PASSWORD parameters.

Example of the GROUP Parameter

```
//TEST  JOB  'D83,123456',GROUP=MYGROUP,USER=MYNAME,PASSWORD=ABC
```

This statement requests that the system connect RACF-defined user MYNAME to the group named MYGROUP for the duration of the job.

MSGCLASS Parameter

Parameter Type: Keyword, optional

Purpose: Use the MSGCLASS parameter to assign the job log to an output class. The job log is a record of job-related information for the programmer. Depending on the JOB statement MSGLEVEL parameter, the job log can consist of:

- Only the JOB statement.
- All job control statements.
- In-stream and cataloged procedure statements.
- Job control statement messages.
- JES and operator messages about the job.

Syntax:

```
MSGCLASS=class
```

Subparameter Definition

class
Identifies the output class for the job log. The class is one character, A through Z or 0 through 9, and must be a valid output class specified at JES initialization.

Defaults

The default is based on the source of the job: The system places the job log in the same output class as the installation-specified default class for the particular card reader, work station, or time-sharing user that submitted the job. The installation default is specified at JES initialization.

Significance of Output Classes

To print the job log and any output data sets on the same output listing, code one of the following:

- The same output class in the DD SYSOUT parameter as in the JOB MSGCLASS parameter.
- DD SYSOUT=* to default to the JOB MSGCLASS output class.
- DD SYSOUT=(,) to default to one of the following:
  1. The CLASS parameter in an explicitly or implicitly referenced OUTPUT JCL statement. In this case, the OUTPUT JCL CLASS parameter should specify the same output class as the JOB MSGCLASS parameter.
  2. The JOB MSGCLASS output class, if no OUTPUT JCL statement is referenced or if the referenced OUTPUT JCL statement contains CLASS=*.

Examples of the MSGCLASS Parameter

```
//EXMP1 JOB   ,GEORGE,MSGCLASS=F
```

In this example, the JOB statement specifies output class F for the job log.

```
//EXMP2 JOB   ,MENTLE,MSGLEVEL=(2,0)
```

This JOB statement does not specify an output class. In this case, the output class defaults to the installation default output class for the device from which the job was submitted.

```
//A1403  JOB   ,BLACK,MSGCLASS=L
//STEP1  EXEC PGM=PRINT
//OUTDD1 DD    SYSOUT=L
```

In this example, the JOB statement and sysout DD statement OUTDD1 both specify the same output class. Consequently, the job log and data set OUTDD1 are written on the same output listing.

```
//B209   JOB    ,WHITE,MSGCLASS=M
//STEPA  EXEC   PGM=PRINT
//OUTDDX DD     SYSOUT=*
```

In this example, the JOB statement specifies that the system route the job log to output class M. The system also routes sysout data set OUTDDX to class M because SYSOUT=* is specified.

MSGLEVEL Parameter

Parameter Type: Keyword, optional

Purpose: Use the MSGLEVEL parameter to control listing of the job log. You can request that the system print the following:

- Only the JOB statement.
- All job control statements in the input stream, that is, all JCL statements and JES2 or JES3 statements.
- In-stream and cataloged procedure statements for any procedure a job step calls.
- Messages about job control statements.
- JES and operator messages about the job's processing: allocation of devices and volumes, execution and termination of job steps and the job, and disposition of data sets.

Syntax:

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

You can omit the parentheses if you code only the first subparameter.

Subparameter Definition

statements

Indicates which job control statements the system is to print in the job log. statements is one of the following numbers:

0 The system prints only the JOB statement.

1 The system prints all JCL statements, JES2 or JES3 control statements, the procedure statements, and IEF653I messages, which give the values assigned to symbolic parameters in the procedure statements.

2 The system prints only JCL statements and JES2 or JES3 control statements.

messages

Indicates which messages the system is to print in the job log. messages is one of the following numbers:

0 The system prints only JCL messages. It prints JES and operator messages only if the job abnormally terminates, and prints SMS messages only if SMS fails the job.

1 The system prints JCL, JES, operator, and SMS messages.

JOB: MSGLEVEL

Defaults

If you do not code the MSGLEVEL parameter, JES uses an installation default specified at initialization.

Examples of the MSGLEVEL Parameter

```
//EXMP3 JOB  ,GEORGE,MSGLEVEL=(2,1)
```

In this example, the JOB statement requests that the system print JCL statements, JCL messages, JES and operator messages, and SMS messages.

```
//EXMP4 JOB  ,MENTLE,MSGLEVEL=0
```

In this example, the JOB statement requests that the system print only the JOB statement and that JES is to use the installation default for messages.

```
//EXMP5 JOB  ,MIKE,MSGLEVEL=(,0)
```

In this example, the JOB statement requests that JES use the installation default for printing JCL statements and the system is not to print JES and operator messages unless the job abnormally terminates. SMS messages are printed only if SMS fails the job.

NOTIFY Parameter

Parameter Type: Keyword, optional

Purpose: Use the NOTIFY parameter to request that the system send a message to your TSO userid or another TSO userid when this background job completes processing.

Syntax:

```
NOTIFY=userid
```

Subparameter Definition

userid
Identifies the user that the system is to notify. The userid is 1 through 7 alphanumeric characters and must be a valid TSO userid.

Relationship to JES2 /*JOBPARM SYSAFF Parameter

If you submit a TSO job with a JOB statement NOTIFY parameter or the job includes a JES2 /*NOTIFY statement, do not specify IND in a JES2 /*JOBPARM SYSAFF parameter; the IND subparameter would change the mode of the job. This change would be invalid because the mode of the job must match the mode of the system at which the job is submitted.

Receiving Notification of Job Completion

In a JES2 System: If you are logged on to the member of the JES2 multi-access spool from which you submitted the job, the system immediately notifies you when the job completes. If you are not logged on, the system saves the message until you log on to the member from which you originally submitted the job.

In a JES3 System: If you are logged on, the system immediately notifies you when the job completes. If you are not logged on, the system saves the message until you log on to the system from which you originally submitted the job.

To receive notification that a job you submitted through batch processing has completed, code an ACMAIN parameter on a JES3 //*MAIN statement in addition to the JOB statement NOTIFY parameter. The ACMAIN parameter names the processor on which your TSO system is running.

If a //*ROUTE or XMIT JCL statement follows the JOB statement, you may not be notified when the transmitted job completes.

Example of the NOTIFY Parameter

```
//SIGN  JOB  ,JEEVES,NOTIFY=POK1
```

When the job SIGN completes processing, the system sends a message to userid POK1.

PASSWORD Parameter

Parameter Type: Keyword, optional

Purpose: Use the PASSWORD parameter to identify a current RACF password or specify a new RACF password. You can specify a new password at any time and must specify a new password when your current one expires.

If the installation specified JES early user verification in its RACF options and contains the user exit routine needed to verify the password, a new password specified in the PASSWORD parameter takes effect when the job is read in. The new password takes effect even if the job is held for execution later and may take effect even if the job fails because of JCL errors. When changing the password, other jobs that use the new or old password may fail, depending on when their passwords are verified.

If the installation contains the feature for propagation of the user and group identification, the USER, the PASSWORD, and, optionally, the GROUP parameters are required on JOB statements only for the following:

- Batch jobs submitted through an input stream, such as a card reader, (1) if the job requires access to RACF-protected resources or (2) if the installation requires that all jobs have RACF identification.
- Jobs submitted by one TSO user for another user. In this case, the JOB statement must specify the other user's userid and password. The group id is optional.
- Jobs that execute at another network node that uses RACF protection.

Otherwise, the USER, PASSWORD, and GROUP parameters can be omitted from JOB statements. RACF uses the userid, password, and default group id of the submitting TSO user or job.

References: For more information on using RACF-protected facilities, see *Resource Access Control Facility (RACF) General Information Manual.*

Syntax:

```
PASSWORD=(password[,new-password])
```

- You can omit the parentheses if you code only the first subparameter.
- The PASSWORD parameter must be on the first statement if the JOB statement is continued.

JOB: PASSWORD

Subparameter Definition

password
Specifies the user's current RACF password. The password is 1 through 8 alphanumeric or national ($, #, @) characters.

new-password
Specifies the user's new RACF password. The new-password is 1 through 8 alphanumeric or national ($, #, @) characters. The installation's security administrator can impose additional restrictions on passwords; follow your installation's rules.

Relationship to Other Parameters

If the installation does **not** contain the user and group identification propagation feature:

- Code a PASSWORD parameter when coding a USER or GROUP parameter on a JOB statement.
- Code a USER parameter when coding a PASSWORD parameter.

Examples of the PASSWORD Parameter

```
//TEST1  JOB  'D83,123456',PASSWORD=ABCDE,USER=MYNAME
```

This JOB statement identifies ABCDE as the current password for the RACF user.

```
//TEST2  JOB 'D83,123456',PASSWORD=(BCH,A12),USER=RAC1,GROUP=GRP1
```

This JOB statement requests that the system change the RACF password from BCH to A12.

PERFORM Parameter

Parameter Type: Keyword, optional

Purpose: Use the PERFORM parameter to specify the performance group for the job. The installation-defined performance groups determine the rate at which associated jobs have access to the processor, storage, and channels.

Syntax:

```
PERFORM=n
```

Subparameter Definition

n

Requests a performance group. The n is a number from 1 through 999 and must identify a performance group that has been defined by your installation. The specified performance group should be appropriate for your job type according to your installation's rules.

Defaults

If no PERFORM parameter is specified or if the specified PERFORM number fails validity checks, the system uses an installation default specified at initialization. If the installation did not specify a default, the system uses a built-in default:

Default	Use
1	For non-TSO jobs
2	For TSO sessions

See *SPL: Initialization and Tuning* for details.

Overrides

A JOB statement PERFORM parameter applies to all steps of the job and overrides any EXEC statement PERFORM parameters.

Code EXEC statement PERFORM parameters when each job step executes in a different performance group. The system uses an EXEC statement PERFORM parameter only when no PERFORM parameter is on the JOB statement and only during the job step.

Example of the PERFORM Parameter

```
//STEP1  JOB  ,MARLA,CLASS=D,PERFORM=25
```

In this example, CLASS=D determines the class in which the system will execute the job. Once in the system, the job will run in performance group 25. The installation must have defined the significance of this performance group.

Programmer's Name Parameter

Parameter Type: Positional, required (according to installation procedures)

Purpose: Use the programmer's name parameter to identify the person or group responsible for a job.

Syntax:

```
programmer's-name
```

Location: Place the programmer's name parameter immediately after the accounting information parameter and before all keyword parameters.

Omission: Do not code a comma to indicate the absence of the programmer's name parameter. For example:

```
//JOBA  JOB  'D58/706',MSGCLASS=A
```

Special Characters: Enclose the programmer's name in apostrophes when:

- The name contains special characters, other than hyphens, leading periods, or embedded periods. For example:

```
//JOBB JOB  ,S-M-TU
//JOBC JOB  ,.ABC
//JOBD JOB  ,P.F.M
//JOBE JOB  ,'BUILD/PAUL'
//JOBF JOB  ,'MAE BIRDSALL'
```

- The last character of the name is a period. For example:

```
//JOBG JOB  ,'A.B.C.'
```

- Code each apostrophe that is part of the name as two consecutive apostrophes. For example, code O'DONNELL as 'O''DONNELL'.

Parameter Definition

programmer's-name
Identifies the job's owner. The name must not exceed 20 characters, including all special characters.

Examples of the Programmer's Name Parameter

```
//APP  JOB  ,G.M.HILL
```

This JOB statement specifies a programmer's name with no accounting information. The leading comma may be optional; check with your installation.

```
//DELTA   JOB  'T.O''NEILL'
```

The programmer's name contains special characters. The installation requires no accounting information. The imbedded apostrophe is coded as two consecutive apostrophes; the entire name must be enclosed in apostrophes.

```
//#308  JOB  (846349,GROUP12),MATTHEW
```

This JOB statement specifies an account number, additional accounting information, and a programmer's name.

```
//JOBA   JOB  'DEPT. 15E'
```

This installation requires the department number in the programmer's name parameter.

PRTY Parameter

Parameter Type: Keyword, optional

Purpose: Use the PRTY parameter to assign a selection priority to your job. Within a JES2 job class or a JES3 job class group, the system selects jobs for execution in order by priority. A job with a higher priority is selected for execution sooner; jobs with the same priority are selected on a first-in first-out basis.

Note: Depending on the JES2 initialization options in use at your installation, JES2 may ignore the PRTY parameter.

References: For more information about priority, see *SPL: JES2 Initialization and Tuning.*

Syntax:

```
PRTY=priority
```

Subparameter Definition

priority
Requests a priority for the job. The priority is a number from 0 through 15 for JES2 and from 0 through 14 for JES3. The highest priority is 15 or 14.

Follow your installation's rules in coding a priority.

Defaults

JES2 determines the job priority from the following, in override order:

1. A JES2 /*PRIORITY statement.
2. A PRTY parameter on the JOB statement.
3. A value calculated from the accounting information on a JES2 /*JOBPARM statement or the JOB statement.
4. An installation default specified at JES2 initialization.

JES3 determines the job priority from the following, in override order:

1. A PRTY parameter on the JOB statement. If the specified priority is invalid, JES3 issues an error message.
2. An installation default specified at JES3 initialization.

Overrides

To assign a different priority to a particular step in the job, code the DPRTY parameter on the EXEC statement for that step. The job's priority applies to any step without a DPRTY parameter.

Example of the PRTY Parameter

```
//JOBA  JOB  1,'JIM WEBSTER',PRTY=12
```

This job has a priority of 12.

RD Parameter

Parameter Type: Keyword, optional

Purpose: Use the RD (restart definition) parameter to:

- Request that the operator perform automatic step restart if the job fails.
- Suppress, partially or totally, the action of the assembler language CHKPT macro instruction or the DD statement CHKPT parameter.

The system can perform automatic restart only if all of the following are true:

- The JOB or EXEC statement contains RD=R or RD=RNC.
- The step to be restarted returned a completion code that indicated no error in the step.
- The operator authorizes a restart.
- The job has a job journal.

A job journal is a sequential data set that contains job-related control blocks needed for restart.

For JES2, specify a job journal by one of the following:

- An installation option during JES2 initialization.
- RD=R or RD=RNC on either the JOB statement or any one EXEC statement in the job.

For JES3, specify a job journal by one of the following:

- An installation option during JES3 initialization.
- RD=R or RD=RNC on either the JOB statement or any one EXEC statement in the job.
- JOURNAL=YES on a JES3 //*MAIN statement in the job.

References: For detailed information on deferred checkpoint restart, see *Checkpoint/Restart User's Guide*.

Syntax:

Subparameter Definition

R (Restart, Checkpoints Allowed)
Indicates that the operator is to perform automatic step restart if the job fails.

RD=R does not suppress checkpoint restarts:

- If the processing program executed in a job step does not include a CHKPT macro instruction, RD=R allows the system to restart execution at the beginning of the abnormally terminated step.
- If the program includes a CHKPT macro instruction, RD=R allows the system to restart execution at the beginning of the step, if the step abnormally terminates before the CHKPT macro instruction is executed.
- If the step abnormally terminates after the CHKPT macro instruction is executed, only checkpoint restart can occur. If you cancel the affects of the CHKPT macro instruction before the system performs a checkpoint restart, the request for automatic step restart is again in effect.

RNC (Restart, No Checkpoints)
Indicates that the operator is to perform automatic step restart if the job fails.

RD=RNC suppresses automatic and deferred checkpoint restarts. It suppresses:

- Any CHKPT macro instruction in the processing program: That is, the operator is not to perform an automatic checkpoint restart, and the system is not to perform a deferred checkpoint restart if the job is resubmitted.
- The DD statement CHKPT parameter.
- The checkpoint at end-of-volume (EOV) facility.

NR (No Automatic Restart, Checkpoints Allowed)
Indicates that the operator is **not** to perform automatic step restart if the job fails.

RD=NR suppresses automatic checkpoint restart but permits deferred checkpoint restarts. It permits:

- A CHKPT macro instruction to establish a checkpoint.
- The job to be resubmitted for restart at the checkpoint. On the JOB statement when resubmitting the job, specify the checkpoint in the RESTART parameter.

If the system fails, RD=NR does not prevent the job from restarting.

NC (No Automatic Restart, No Checkpoints)
Indicates that the operator is **not** to perform automatic step restart if the job fails.

RD=NC suppresses automatic and deferred checkpoint restarts. It suppresses:

- Any CHKPT macro instruction in the processing program.
- The DD statement CHKPT parameter.
- The checkpoint at EOV facility.

JOB: RD

Defaults

If you do not code the RD parameter, the system uses the installation default from the job's job class specified at initialization.

Overrides

A JOB statement RD parameter applies to all steps of the job and overrides any EXEC statement RD parameters.

Code EXEC statement RD parameters when each job step requires different restart types. The system uses an EXEC statement RD parameter only when no RD parameter is on the JOB statement and only during the job step.

Relationship to Other Control Statements

RD=NC or RD=RNC suppresses the action of the DD statement CHKPT parameter.

Examples of the RD Parameter

```
//JILL  JOB  333,TOM,RD=R
```

RD=R specifies that the operator is to perform automatic step restart if the job fails.

```
//TRY56  JOB  333,DICK,RD=RNC
```

RD=RNC specifies that, if the job fails, the operator is to perform automatic step restart beginning with the step that abnormally terminates. RD=RNC suppresses automatic and deferred checkpoint restarts.

```
//PASS  JOB  (721,994),HARRY,RD=NR
```

RD=NR specifies that the operator is not to perform automatic step restart or automatic checkpoint restart. However, a CHKPT macro instruction can establish checkpoints to be used later for a deferred restart.

REGION Parameter

Parameter Type: Keyword, optional

Purpose: Use the REGION parameter to specify the amount of space that the job requires.

The specified or default region size sets an upper boundary to limit region size for variable-length GETMAINs. The system uses the upper boundary for variable-length GETMAINs as long as the region still has available at least the minimum amount of storage requested.

In addition, the IBM- or installation-supplied routine IEALIMIT or IEFUSI uses the region size to establish a second limiting value. 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 minimum requested.

If the minimum requested length for variable-length GETMAINs or the amount requested for a fixed-length GETMAIN exceeds this second value, the job or job step abnormally terminates.

Specifying a Region Size Value: If your installation does not change the IBM-supplied default limits in the IEALIMIT and/or IEFUSI exit routine modules, then specifying various values for the region size have the following results:

- A value equal to 0K or 0M -- gives the job all the storage available below and above 16 megabytes. The resulting size of the region below and above 16 megabytes is unpredictable.
- A value greater than 0K or 0M and less than or equal to 16384K or 16M -- establishes the size of the private area below 16 megabytes. If the region size specified is not available below 16 megabytes, the job abnormally terminates. The extended region size is the default value of 32 megabytes.
- A value greater than 16384K or 16M and less than or equal to 32768K or 32M -- gives the job all the storage available below 16 megabytes. The resulting size of the region below 16 megabytes is unpredictable. The extended region size is the default value of 32 megabytes.
- A value greater than 32768K or 32M and less than or equal to 2096128K or 2047M -- gives the job all the storage available below 16 megabytes. The resulting size of the region below 16 megabytes is unpredictable. The extended region size is the specified value. If the region size specified is not available above 16 megabytes, the job abnormally terminates.

References: For more information on the region size, see *SPL: System Modifications*. For more information on region size with checkpoint/restart jobs, see *Checkpoint/Restart User's Guide*.

Syntax:

```
REGION=(valueK)
       (valueM)
```

JOB: REGION

Subparameter Definition

valueK
Specifies the required storage in kilobytes (1 kilobyte = 1024 bytes). The value is 1 through 7 decimal numbers, from 1 through 2096128. Code an even number. For example, REGION=66K. If you code an odd number, the system treats it as the next highest even number.

valueM
Specifies the required storage in megabytes (1 megabyte = 1024 kilobytes). The value is 1 through 4 decimal numbers, from 1 through 2047. Code either an even or odd number. For example, REGION=3M.

Defaults

If no REGION parameter is specified, the system uses an installation default specified at JES initialization.

Overrides

A JOB statement REGION parameter applies to all steps of the job and overrides any EXEC statement REGION parameters.

Code EXEC statement REGION parameters when each job step requires a different region size. The system uses an EXEC statement REGION parameter only when no REGION parameter is on the JOB statement and only during the job step.

Relationship to the JOB ADDRSPC Parameter

When ADDRSPC=REAL: Code a REGION parameter to specify how much real storage the job needs. If you omit the REGION parameter, the system uses the default.

When ADDRSPC=VIRT or ADDRSPC is Omitted: Code a REGION parameter to specify how much virtual storage the job needs. If you omit the REGION parameter, the system uses the default.

Examples of the REGION Parameter

```
//ACCT1  JOB  A23,SMITH,REGION=100K,ADDRSPC=REAL
```

This JOB statement indicates that the job requires 100K of real storage.

```
//ACCT4  JOB  175,FRED,REGION=250K
```

This JOB statement indicates that the job requires 250K of virtual storage. When the ADDRSPC parameter is omitted, the system defaults to ADDRSPC=VIRT.

RESTART Parameter

Parameter Type: Keyword, optional

Purpose: Use the RESTART parameter to restart a job. You can specify that the system perform either of two restarts:

- **Deferred step restart**, which is a restart at the beginning of a job step.
- **Deferred checkpoint restart**, which is a restart from a checkpoint taken during step execution by a CHKPT macro instruction.

References: For detailed information on the deferred checkpoint restart, see *Checkpoint/Restart User's Guide*.

Syntax:

```
RESTART=(  { *                    }  [,checkid]  )
           { stepname             }
           { stepname.procstepname}
```

You can omit the parentheses if you code only the first subparameter.

Subparameter Definition

*****

Indicates that the system is to restart execution (1) at the beginning of or within the first job step or (2), if the first job step calls a cataloged or in-stream procedure, at the beginning of or within the first procedure step.

stepname

Indicates that the system is to restart execution at the beginning of or within a job step. If stepname refers to an EXEC statement that invokes a procedure, the step name of the step within the procedure must also be specified.

stepname.procstepname

Indicates that the system is to restart execution at the beginning of or within a step of a cataloged procedure. Stepname identifies the EXEC statement of the job step that calls the procedure; procstepname identifies the EXEC statement of the procedure step.

checkid

Specifies the name of the checkpoint at which the system is to restart execution. This checkpoint must be in the job step specified in the first subparameter.

Omit checkid to request restart at the beginning of the specified job step.

When the name contains special characters, enclose it in apostrophes. Code each apostrophe that is part of the name as two consecutive apostrophes. For example, code CHPT'1 as 'CHPT''1'.

JOB: RESTART

Relationship to Other Control Statements

When the system is to restart execution in a job step, place a SYSCHK DD statement immediately following the JOB statement. The SYSCHK DD statement defines the data set on which the system entered the checkpoint for the step being restarted.

When preparing for a deferred checkpoint, code the DISP abnormal termination disposition subparameter in the step's DD statements as follows:

- KEEP, to keep all data sets that the restart step is to use.
- CATLG, to catalog all data sets that you are passing from steps preceding the restart step to steps following the restart step.

In JES2 systems, you can also use the RESTART parameter on the /*JOBPARM control statement.

In JES3 systems, use can also use the FAILURE parameter on the //*MAIN control statement.

Cautions when Coding the RESTART Parameter

Before resubmitting a job:

- Check all backward references to steps before the restart step. Eliminate all backward references in EXEC statement PGM parameters and DD statement VOLUME=REF parameters.
- Review all EXEC statement COND parameters. If any of the COND parameters reference a step before the restart step, be aware that the system ignores the return code tests for those steps.

Generation Data Sets in Restarted Jobs

In the restart step or following steps, do not use the original relative generation numbers to refer to generation data sets that were created and cataloged before the restart step. Instead, refer to a generation data set by its present relative generation number.

For example, if the last generation data set created and cataloged was assigned a generation number of +2, refer to it as 0 in the restart step and following steps. If generation data set +1 was also created and cataloged, refer to it as -1.

If generation data sets created in the restart step were kept instead of cataloged, that is, DISP=(NEW,CATLG,KEEP) was coded, then refer to them by the same relative generation numbers used to create them.

Examples of the RESTART Parameter

```
//LINES  JOB  '1/17/85',RESTART=COUNT
```

This JOB statement indicates that the system is to restart execution at the beginning of the job step named COUNT.

```
//@LOC5  JOB  '4/11/86',RESTART=(PROCESS,CHKPT3)
//SYSCHK DD   DSNAME=CHK,UNIT=3330,DISP=OLD
```

The JOB statement indicates that the system is to restart execution at checkpoint CHKPT3 in job step PROCESS. The SYSCHK DD statement must follow the JOB statement; it defines the data set on which the system wrote checkpoint CHKPT3.

```
//WORK  JOB  ,PORTER,RESTART=(*,CKPT2)
//SYSCHK DD  DSNAME=CHKPT,UNIT=3330,DISP=OLD
```

The JOB statement indicates that the system is to restart execution at checkpoint CKPT2 in the first job step. The SYSCHK DD statement defines the data set on which the system wrote checkpoint CKPT2.

```
//CLIP5  JOB  ,COLLINS,RESTART=(PAY.WEEKLY,CHECK8)
//SYSCHK DD   DSNAME=CHKPT,UNIT=3350,DISP=OLD
```

The JOB statement indicates that the system is to restart execution at checkpoint CHECK8 in procedure step WEEKLY. PAY is the name field on the EXEC statement that calls the cataloged procedure that contains procedure step WEEKLY. The SYSCHK DD statement defines the data set on which the system wrote checkpoint CHECK8.

TIME Parameter

Parameter Type: Keyword, optional

Purpose: Use the TIME parameter to specify the maximum length of time that a job is to use the processor and to find out through messages how much processor time the job used.

The system terminates a job that exceeds the specified time limit unless a user exit routine extends the time. The exit routine is established through System Management Facilities (SMF).

References: See *SPL: System Management Facilities (SMF).*

Syntax:

```
TIME={([minutes][,seconds])}
     {1440                 }
```

You can omit the parentheses if you code only 1440 or the processor time in minutes.

Subparameter Definition

minutes
Specifies the maximum number of minutes the job can use the processor. The minutes must be a number from 1 through 1439.

Do not code TIME=0 or TIME=(,) on the JOB statement. The results are unpredictable.

seconds
Specifies the maximum number of seconds that the job can use the processor, in addition to any minutes that are specified. The seconds must be a number from 1 through 59.

1440
Indicates that the job can use the processor for an unlimited amount of time; 1440 literally means 24 hours. Code TIME=1440 for the following reasons:

- To obtain job accounting information.
- To specify that the system is to allow any of the job's steps to remain in a continuous wait state for more than the installation time limit, which is established through SMF.

Defaults

If no JOB TIME parameter is specified, JES terminates the job when a job step exceeds its maximum time limit.

If 1440 is not specified, SMF uses its current job wait time limit.

Overrides

For a JOB statement TIME parameter other than TIME=1440, the system sets the time limit for each step to:

- The step time limit specified on the EXEC statement TIME parameter or the job time remaining after execution of previous job steps, whichever is smaller.
- If no EXEC TIME parameter was specified, (1) the default time limit or (2) the job time remaining after execution of previous steps, whichever is smaller.

Time Handling

How the System Converts the Time Value: The job time limit or the time remaining after execution of previous steps in a job is converted by the system to seconds and then rounded to the nearest unit, where 1 unit = 1.048576 seconds. Thus, a step can begin execution with up to one-half unit more or one-half unit less time than expected. If the time remaining for the job is less than one-half unit, a step will begin execution with zero time, resulting in an abnormal termination.

Time Checking: Because the system checks the processor time-used field about every 10.5 seconds, the actual time that a job uses the processor can exceed the specified TIME value by up to 10.5 seconds. For example, the system checks the job's time-used field and finds 0.5 seconds remaining. Because the system does not again check the job's time-used field for about 10.5 seconds, the job can execute for an additional 10.5 seconds and thus exceed the coded TIME value by 10 seconds.

Examples of the TIME Parameter

```
//STD1  JOB  ACCT271,TIME=(12,10)
```

This statement specifies that the maximum amount of time the job can use the processor is 12 minutes, 10 seconds.

```
//TYPE41 JOB  ,GORDON,TIME=(,30)
```

This statement specifies that the maximum amount of time the job can use the processor is 30 seconds.

```
//FORMS  JOB  ,MORRILL,TIME=5
```

This statement specifies that the maximum amount of time the job can use the processor is 5 minutes.

```
//RAINCK JOB  374231,MORRISON,TIME=1440
```

This statement specifies an unlimited amount of time for job execution; the job can use the processor and remain in wait state for an unspecified period of time. The system will issue messages telling how much processor time the job used.

Examples of the TIME Parameter on JOB and EXEC Statements

```
//FIRST JOB      ,SMITH,TIME=2
//STEP1 EXEC     PGM=READER,TIME=1
         .
         .
         .
//STEP2 EXEC     PGM=WRITER,TIME=1
         .
```

In this example, the job is allowed 2 minutes for execution and each step is allowed 1 minute. If either step continues executing beyond 1 minute, the entire job abnormally terminates beginning with that step.

```
//SECOND JOB      ,JONES,TIME=3
//STEP1  EXEC     PGM=ADDER,TIME=2
          .
          .
          .
//STEP2  EXEC     PGM=PRINT,TIME=2
          .
```

In this example, the job is allowed 3 minutes for execution, and each step is allowed 2 minutes. If either step continues executing beyond 2 minutes, the entire job abnormally terminates beginning with that step. If STEP1 executes for 1.74 minutes and STEP2 tries to execute beyond 1.26 minutes, the job abnormally terminates because of the 3-minute limit specified on the JOB statement.

TYPRUN Parameter

Parameter Type: Keyword, optional

Purpose: Use the TYPRUN parameter to request special job processing. The TYPRUN parameter can tell the system to:

- In a JES2 system, copy the input job stream directly to a sysout data set and schedule it for output processing.
- In a JES2 or JES3 system, place a job on hold until a special event occurs. When the event occurs, the operator, following your directions, must release the job from its hold to allow the system to select the job for processing. Use the JES2 /*MESSAGE statement or the JES3 //*OPERATOR statement to notify the operator to release the job.
- In a JES2 or JES3 system, scan a job's JCL for syntax errors.

Syntax:

```
TYPRUN=(COPY   )
       |HOLD   |
       {JCLHOLD}
       (SCAN   )
```

Subparameter Definition

COPY (JES2 only)
Requests that JES2 copy the input job stream, as submitted, directly to a sysout data set and schedule the sysout data set for output processing. The system does not schedule the job for execution. The class of this sysout data set is the same as the message class of the job and is controlled by the JOB MSGCLASS parameter.

Note: COPY is supported only in JES2 systems.

HOLD
Requests that the system hold the job before execution until the operator releases it. The operator should release the job when a particular event occurs. If an error occurs during input service processing, JES does not hold the job.

JCLHOLD (JES2 only)
Requests that JES2 hold the job before completing JCL processing. JES2 holds the job until the operator releases it.

Note: JCLHOLD is supported only in JES2 systems.

SCAN

Requests that the system scan this job's JCL for syntax errors, without executing the job or allocating devices. This parameter asks the system to check for:

- Invalid spelling of parameter keywords and some subparameter keywords.
- Invalid characters.
- Unbalanced parentheses.
- Misplaced positional parameters on some statements.
- In a JES3 system only, parameter value errors or excessive parameters.
- Invalid syntax on JCL statements in cataloged procedures invoked by any scanned EXEC statements.

The system does not check for misplaced statements, for invalid syntax in JCL subparameters, or for parameters and/or subparameters that are inappropriate together.

In a JES3 system, the system does not scan the JCL on the submitting system when a //*ROUTE or XMIT JCL statement follows the JOB statement.

Relationship to Other Control Statements

In a JES3 system, code PGM=JCLTEST or PGM=JSTTEST on the EXEC statement to scan a job step's JCL. JCLTEST or JSTTEST provides for a step the same function as provided by TYPRUN=SCAN for a job.

Example of the TYPRUN Parameter

```
//UPDATE JOB    ,HUBBARD
//STEP1  EXEC   PGM=LIBUTIL
          .
          .
          .
//LIST   JOB    ,HUBBARD,TYPRUN=HOLD
//STEPA  EXEC   PGM=LIBLIST
          .
          .
          .
```

Jobs UPDATE and LIST are submitted for execution in the same input stream. UPDATE executes a program that adds and deletes members of a library; LIST executes a program that lists the members of that library. For an up-to-date listing of the library, LIST must execute after UPDATE. To force this execution order, code TYPRUN=HOLD on JOB statement LIST.

If a MONITOR JOBNAMES command is executed from the input stream or by the operator, the system notifies the console operator when UPDATE completes. The operator can then release LIST, allowing the system to select LIST for execution.

USER Parameter

Parameter Type: Keyword, optional

Purpose: Code the USER parameter to identify to the system the person submitting the job. The userid is used by the Resource Access Control Facility (RACF), the system resources manager (SRM), and other system components.

If the installation contains the feature for propagation of the user and group identification, the USER, the PASSWORD, and, optionally, the GROUP parameters are required on JOB statements only for the following:

- Batch jobs submitted through an input stream, such as a card reader, (1) if the job requires access to RACF-protected resources or (2) if the installation requires that all jobs have RACF identification.
- Jobs submitted by one TSO user for another user. In this case, the JOB statement must specify the other user's userid and password. The group id is optional.
- Jobs that execute at another network node that uses RACF protection.

Otherwise, the USER, PASSWORD, and GROUP parameters can be omitted from JOB statements. RACF uses the userid, password, and default group id of the submitting TSO user or job.

References: For more information on RACF-protected facilities, see *Resource Access Control Facility (RACF) General Information Manual.*

Syntax:

```
USER=userid
```

Subparameter Definition

userid
Identifies a user to the system. The userid consists of 1 through 7 alphanumeric or national ($, #, @) characters; the first character must be alphabetic or national ($, #, @).

Defaults

If neither the JOB statement nor the submitting TSO user supplies identification information, RACF assigns a default userid and group id, unless the job enters the system via a JES internal reader. In that case, the user and default group identification of the submitting TSO user or job is used.

JOB: USER

Relationship to Other Parameters

If the JOB statement contains a GROUP or PASSWORD parameter, the statement must also contain a USER parameter.

Example of the USER Parameter

```
//TEST  JOB  'D83,123456',USER=MYNAME,PASSWORD=ABCD
```

This statement identifies the user submitting this job as MYNAME.

Chapter 16. Null Statement

Use the null statement to mark the end of a job.

Syntax:

//
• The null statement consists of the characters // in columns 1 and 2. • The rest of the statement **must** be blank.

Location in the JCL

Place a null statement (1) at the end of a job's control statements and data and (2) at the end of an input stream.

The system can also recognize the end of a job when it reads the next JOB statement or when the input stream contains no more records.

A null statement that does not end an input stream should be immediately followed by a JOB statement. The system ignores statements between a null statement and the next valid JOB statement.

Note: JES2 ignores a NULL statement when it is included in a job's JCL statements. JES2 processes JES2 control statements following a NULL statement as part of the job (until the next JOB statement or EOF).

If a null statement follows a control statement that is being continued, the system treats the null statement as a blank comment field and assumes that the control statement contains no other parameters.

Null Statement

Example of the Null Statement

```
//MYJOB JOB    ,'C BROWN'
//STEP1 EXEC   PROC=FIELD
//STEP2 EXEC   PGM=XTRA
//DD1   DD     UNIT=3400-5
//DD2   DD     *
        .
        .
        data
        .
/*
//
```

The null statement indicates the end of job **MYJOB**.

Chapter 17. OUTPUT JCL Statement

Purpose: Use the OUTPUT JCL statement to specify processing options for a system output (sysout) data set. These processing options are used only when the OUTPUT JCL statement is explicitly or implicitly referenced by a sysout DD statement. JES combines the options from this OUTPUT JCL statement with the options from the referencing DD statement.

OUTPUT JCL statements are useful in processing the output of one sysout data set in several ways. For example, a sysout data set can be sent to a distant site for printing, as shown in statement OUT1, while it is also printed locally, as shown in statement OUT2:

```
//OUT1  OUTPUT  DEST=STLNODE.WMSMITH
//OUT2  OUTPUT  CONTROL=DOUBLE
//DS    DD      SYSOUT=C,OUTPUT=(*.OUT1,*.OUT2)
```

The parameters you can specify for sysout data set processing are arranged alphabetically in the following pages.

Note: For JES3, some values you specify on the OUTPUT JCL statement may be lost when the sysout data set is requeued from the JES3 hold queue to the writer queue. For example, values on the FORMS or COPIES parameter may be lost, in which case the system uses the default values.

References: For information about the JES initialization parameters that provide installation defaults, see *SPL: JES2 Initialization and Tuning* and *SPL: JES3 Initialization and Tuning*.

Syntax:

```
//name  OUTPUT parameter[,parameter]...  [comments]
```

The OUTPUT JCL statement consists of the characters // in columns 1 and 2 and four fields: name, operation (OUTPUT), parameter, and comments.

Name Field

Code a name in the name field of every OUTPUT JCL statement, as follows:

- Each OUTPUT JCL name must be unique within a job.
- The name must begin in column 3.
- The name is 1 through 8 alphanumeric or national ($, #, @) characters.
- The first character must be alphabetic or national ($, #, @).
- The name must be followed by at least one blank.

OUTPUT JCL

Operation Field

The operation field consists of the characters OUTPUT and must be preceded and followed by at least one blank. It can begin in any column.

Parameter Field

The OUTPUT JCL statement contains only keyword parameters. All parameters are optional; however, do not leave the parameter field blank. You can code any of the keyword parameters in any order in the parameter field.

KEYWORD PARAMETERS	VALUES	PURPOSE
BURST= {YES / Y / NO / N} See page 17-9	YES or Y: burster-trimmer-stacker NO or N: continuous forms stacker	Directs output to a stacker on a 3800 Printing Subsystem.
CHARS= {table-name / (table-name[,table-name]...) / STD / DUMP / (DUMP[,table-name]...)} See page 17-11	1 - 4 table-name subparameters: 1 - 4 alphanumeric or $, #, @ characters STD: character-arrangement table (JES3 only) DUMP: 204-character print lines on 3800 dump	Names character-arrangement tables for printing on a 3800 Printing Subsystem. Can request a high-density dump on a SYSABEND or SYSUDUMP DD statement.
CKPTLINE=nnnnn See page 17-14	nnnnn: 0 - 32767	Specifies the maximum lines in a logical page. (JES3 support is limited to 3800 Printing Subsystem Models 3, 6 and 8.)
CKPTPAGE=nnnnn See page 17-15	nnnnn: 1 - 32767	Specifies the number of logical pages to be printed or transmitted before JES takes a checkpoint. (JES3 support is limited to 3800 Printing Subsystem Models 3, 6 and 8.)
CKPTSEC=nnnnn See page 17-16	nnnnn: 1 - 32767	Specifies how many seconds of printing are to elapse between each checkpoint of this sysout data set. (JES3 support is limited to 3800 Printing Subsystem Models 3, 6 and 8.)
CLASS= {class / *} See page 17-17	class: A - Z, 0 - 9 *: same output class as MSGCLASS parameter on JOB statement	Assigns the sysout data set to an output class.

KEYWORD PARAMETERS	VALUES	PURPOSE
COMPACT = compaction-table-name See page 17-20	compaction-table-name: 1 - 8 alphanumeric characters	Specifies a compaction table for sending this sysout data set to a SNA remote terminal.
CONTROL = { PROGRAM | SINGLE | DOUBLE | TRIPLE } See page 17-21	PROGRAM: each logical record begins with a carriage control character SINGLE: single spacing DOUBLE: double spacing TRIPLE: triple spacing	Specifies that the data set records begin with carriage control characters or specifies line spacing.
COPIES = { nnn | (,(group-value[,group-value]...)) } See page 17-22	nnn (JES2): 1 - 255 nnn (JES3): 0 - 255 1 - 8 group-values (JES2): 1 - 255 1 - 8 group values (JES3): 1 - 254	Specifies number of copies printed. For a 3800 Printing Subsystem, can instead specify number of copies of each page printed before the next page is printed.
DATACK = { BLOCK | UNBLOCK | BLKCHAR | BLKPOS } See page 17-25	BLOCK: indicates errors are not reported UNBLOCK: indicates errors are reported BLKCHAR: indicates print errors are blocked BLKPOS: indicates data errors are blocked	Indicates whether or not print-positioning errors and invalid character data-check errors are to be blocked or not blocked.
DEFAULT = { YES | Y | NO | N } See page 17-27	YES or Y: this statement can be implicitly referenced by sysout DD statements NO or N: this statement cannot be implicitly referenced by sysout DD statements.	Specifies that this is a default OUTPUT JCL statement.
DEST = destination destination (JES2): LOCAL name Nnnnn NnnRmmmm to NnnnnRmm nodename.userid Rnnnn or RMnnnn or RMTnnnn Unnnn destination (JES3): ANYLOCAL device-name group-name nodename nodename.remote See page 17-30	LOCAL: local device name: named local or remote device Nnnnn: node (1 - 1000) NnRm: node (1 - 1000) and remote work station (1 - 9999); 6 digits maximum for n and m combined nodename.userid: node (1 - 8 alphanumeric characters) and userid (1 - 8 alphanumeric characters) Rnnnn or RMnnnn or RMTnnnn: remote terminal (1 - 9999) Unnnn: local terminal (1 - 9999) ANYLOCAL: any local device device-name: local device (1 - 8 alphanumeric or $, #, @ characters) group-name: 1 or more local devices or remote stations (1 - 8 alphanumeric or $, #, @ characters) nodename: node (1 - 8 alphanumeric or $, #, @ characters) remote: remote work station (1 - 8 alphanumeric or $, #, @ characters)	Sends a sysout data set to the specified destination.
FCB = { fcb-name | STD } See page 17-33	fcb-name: 1 - 4 alphanumeric or $, #, @ characters STD: standard FCB (JES3 only)	Specifies FCB image, carriage control tape for 1403 Printer, or data-protection image for 3525 Card Punch.

OUTPUT JCL

KEYWORD PARAMETERS	VALUES	PURPOSE
FLASH = { overlay-name / (overlay-name[,count]) / (,count) / NONE / STD } See page 17-35	overlay-name: forms overlay frame (1 - 4 alphanumeric or $, #, @ characters) count: copies with overlay (0 - 255) NONE: suppresses flashing STD: standard forms flash overlay (JES3 only)	For printing on a 3800 Printing Subsystem, indicates that the data set is to be printed with forms overlay and can specify how many copies are to be flashed.
FORMDEF = membername See page 17-38	membername: 1 - 6 alphanumeric or $, #, @ characters	Names a library member that PSF uses in printing the sysout data set on a page-mode printer (such as a 3800 Model 3).
FORMS = { form-name / STD } See page 17-40	form-name: 1 - 8 alphanumeric or $, #, @ characters STD: standard form (JES3 only)	Identifies forms on which the sysout data set is to be printed or punched.
GROUPID = output-group See page 17-41	output-group: 1 - 8 alphanumeric characters	Specifies that this sysout data set belongs to a user-named output group. (JES2 only)
INDEX = nn See page 17-43	nn: 1 - 31	Specifies how many print positions the left margin is to be indented for a sysout data set printed on a 3211 Printer with the indexing feature. (JES2 only)
JESDS = { ALL / JCL / LOG / MSG } See page 17-44	ALL: all of job's system-managed data sets JCL: all JCL processing data sets LOG: job's hard-copy log MSG: job's system messages	Requests that the indicated system-managed data sets for the job be processed according to the parameters on this OUTPUT JCL statement.
LINDEX = nn See page 17-47	nn: 1 - 31	Specifies how many print positions the right margin is to be moved in from the full page width for a sysout data set printed on a 3211 Printer with the indexing feature. (JES2 only)
LINECT = nnn See page 17-48	nnn: 0 - 255	Specifies the maximum lines JES2 is to print on each page. (JES2 only)
MODIFY = { module-name / ([module-name][,trc]) } See page 17-49	module-name: 1 - 4 alphanumeric or $, #, @ characters trc: table-name in CHARS parameter (0 for first, 1 for second, 2 for third, and 3 for fourth table-name)	Specifies a copy-modification module in SYS1.IMAGELIB to be used by JES to print the data set on a 3800 Printing Subsystem.

KEYWORD PARAMETERS	VALUES	PURPOSE
PAGEDEF=membername See page 17-51	membername: 1 - 6 alphanumeric or $, #, @ characters	Names a library member that PSF uses in printing the sysout data set on a page-mode printer (such as a 3800 Model 3).
PIMSG={(YES[,msg-count]) (NO[,msg-count])} See page 17-53	YES: print messages from a functional subsystem NO: not print messages from a functional subsystem msg-count: number of errors to cause printing to be terminated (0 - 999)	Indicates that messages from a functional subsystem should or should not be printed in the listing following the sysout data set.
PRMODE={LINE PAGE process-mode} See page 17-55	LINE: send data set to line-mode printer PAGE: send data set to page-mode printer process-mode: installation-defined mode (1 - 8 alphanumeric characters)	Identifies the process mode required to print the sysout data set.
PRTY=nnn See page 17-57	nnn: 0 - 255 (0 is lowest, 255 is highest)	Specifies initial priority at which the sysout data set enters the output queue.
THRESHLD=limit See page 17-58	limit: 1 - 99999999	Specifies the maximum size for a sysout data set. Use it to obtain simultaneous printing of large data sets or many data sets from one job. (JES3 only)
TRC={YES Y NO N} See page 17-60	YES or Y: data set contains TRC codes NO or N: data set does not contain TRC codes	Specifies whether or not the sysout data set's records contain table reference codes (TRC) as the second character.
UCS=character-set-code See page 17-62	character-set-code: 1 - 4 alphanumeric or $, #, @ characters	Specifies universal character set, print train, or character-arrangement table for a 3800 Printing Subsystem.
WRITER=name See page 17-65	name: 1 - 8 alphanumeric characters	Names an external writer to process the sysout data set rather than JES.

Default OUTPUT JCL Statement: An OUTPUT JCL statement that contains a DEFAULT=YES parameter is called a default OUTPUT JCL statement.

Comments Field

The comments field follows the parameter field after at least one intervening blank.

OUTPUT JCL

Location in the JCL

You must place an OUTPUT JCL statement in the input stream before any sysout DD statement that refers to it.

References by Sysout DD Statements: An OUTPUT JCL statement can be referenced by a sysout DD statement in two ways:

- **Explicitly.** The sysout DD statement contains an OUTPUT parameter that specifies the name of the OUTPUT JCL statement.

- **Implicitly.** The sysout DD statement does not contain an OUTPUT parameter. Implicit references are to default OUTPUT JCL statements and require that the job or step contain one or more default OUTPUT JCL statements preceding the sysout DD statement.

Note: If the sysout DD statement does not contain an OUTPUT parameter and the job or step does not contain a default OUTPUT JCL statement, processing of the sysout data set is controlled only by the DD statement, a JES2 /*OUTPUT statement or JES3 //*FORMAT statement, and appropriate installation defaults.

Job-Level OUTPUT JCL Statements: This statement appears after the JOB statement and before the first EXEC statement.

Step-Level OUTPUT JCL Statements: This statement appears in a step, that is, anywhere after the first EXEC statement in a job.

Location of Default OUTPUT JCL Statements: Where you place default OUTPUT JCL statements determines which statements a sysout DD statement refers to in an implicit reference, as follows:

- A sysout DD statement implicitly references all step-level default OUTPUT JCL statements in the same step.

- A sysout DD statement implicitly references all job-level default OUTPUT JCL statements when the step containing the DD statement does not contain any step-level default OUTPUT JCL statements.

You can place more than one job- or step-level default OUTPUT JCL statement in a job or step.

OUTPUT JCL Statement with JESDS Parameter: Place an OUTPUT JCL statement with a JESDS parameter after the JOB statement and before the first EXEC statement.

OUTPUT JCL Statements in Cataloged or In-Stream Procedures: OUTPUT JCL statements can appear in procedure steps. The referencing DD statement can appear later in the procedure, in the calling job step, or in a later step in the job.

An OUTPUT JCL statement must not be placed before the first EXEC statement in a procedure; for this reason, procedures cannot contain job-level OUTPUT JCL statements or OUTPUT JCL statements with JESDS parameters.

A procedure DD statement can refer to an OUTPUT JCL statement in an earlier job step or to a job-level OUTPUT JCL statement. However, a procedure DD statement cannot refer to an OUTPUT JCL statement in the calling step.

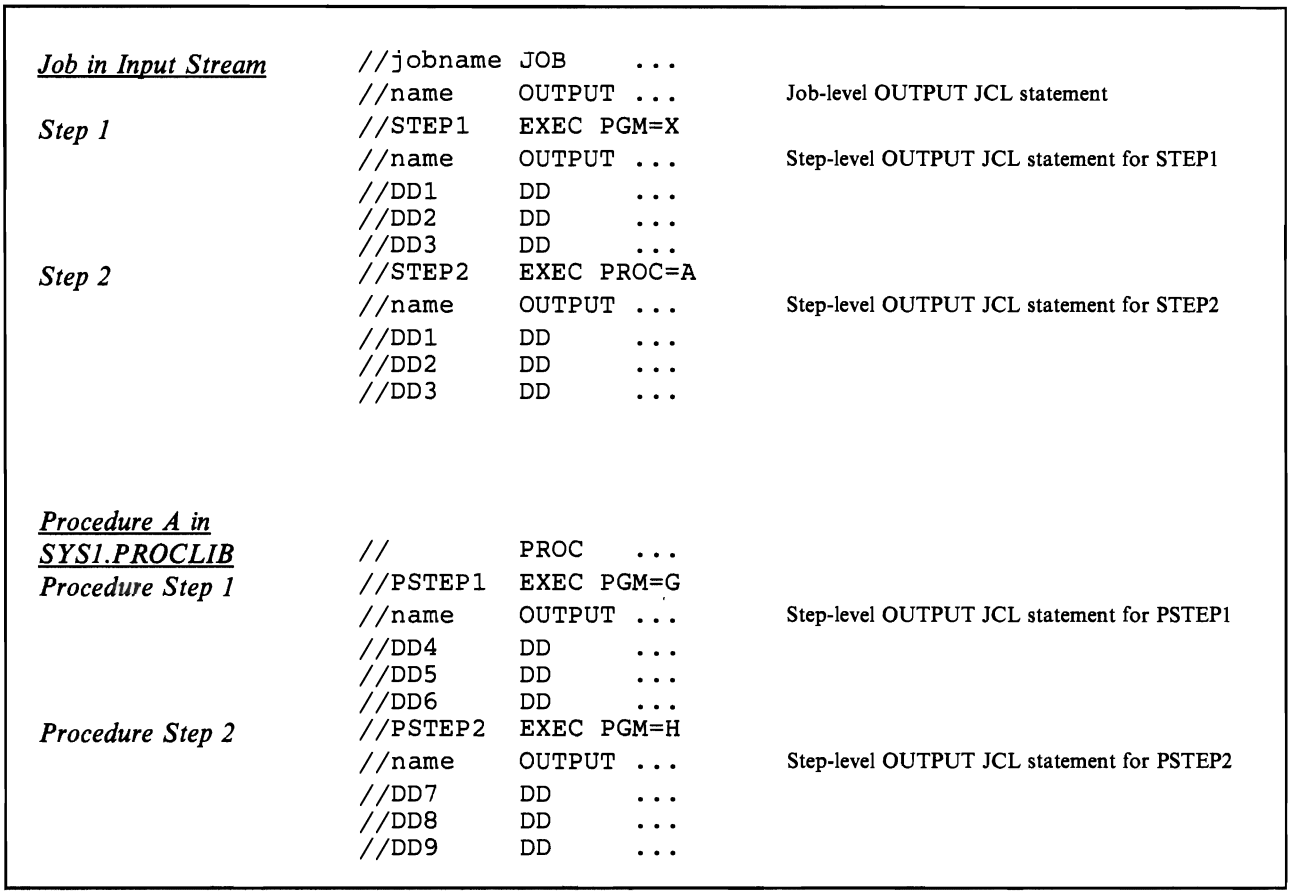


Figure 17-1. Job- and Step-Level OUTPUT JCL Statements in the JCL

Overrides

- Parameters on a sysout DD statement override corresponding parameters on an OUTPUT JCL statement.
- Parameters that appear only on the sysout DD statement or only on the OUTPUT JCL statement are used by JES in processing the data set.

Relationship to Sysout DD Statement

Do not refer to an OUTPUT JCL statement in a sysout DD statement that defines a JES internal reader. Such a DD statement contains an INTRDR subparameter in the SYSOUT parameter.

OUTPUT JCL

Relationship to the JES2 /*OUTPUT Statement

JES2 ignores a JES2 /*OUTPUT statement when either of the following appears in the same job or step:

- A default OUTPUT JCL statement implicitly referenced by the sysout DD statement.
- An OUTPUT JCL statement explicitly referenced by the OUTPUT parameter of the sysout DD statement.

In this case, JES2 uses the third positional subparameter of the DD SYSOUT parameter as a form name, and not as a reference to a JES2 /*OUTPUT statement.

Relationship to the JES3 //*FORMAT Statement

- When a sysout DD statement implicitly or explicitly references an OUTPUT JCL statement, JES3 ignores any default JES3 //*FORMAT statements in the job. A default //*FORMAT statement contains a DDNAME=, parameter.
- When a JES3 //*FORMAT statement contains a DDNAME parameter that explicitly references a sysout DD statement, JES3 ignores any default OUTPUT JCL statements in the job.
- JES3 uses the processing options from both a JES3 //*FORMAT statement and an OUTPUT JCL statement in a job when (1) the //*FORMAT statement DDNAME parameter names a sysout DD statement and (2) the sysout DD statement's OUTPUT parameter names an OUTPUT JCL statement. Two separate sets of output are created from the data set defined by the sysout DD statement:
  - One processed according to the options on the JES3 //*FORMAT statement combined with the sysout DD statement.
  - One processed according to the options on the OUTPUT JCL statement combined with the sysout DD statement.

For more information on the use of the OUTPUT JCL statement with JES3, see *MVS/XA SPL: JES3 Initialization and Tuning*.

BURST Parameter

Parameter Type: Keyword, optional

Purpose: Use the BURST parameter to specify that the output for the sysout data set printed on a 3800 Printing Subsystem is to go to:

- The burster-trimmer-stacker, to be burst into separate sheets.
- The continuous forms stacker, to be left in continuous fanfold.

If the specified stacker is different from the last stacker used, or if a stacker was not previously requested, JES issues a message to the operator to thread the paper into the required stacker.

Note: BURST applies only for a data set printed on a 3800 equipped with a burster-trimmer-stacker.

Syntax:

```
BURST=(YES)
      (Y  )
      (NO )
      (N  )
```

Subparameter Definition

YES
Requests that the printed output is to be burst into separate sheets. This subparameter can also be coded as Y.

NO
Requests that the printed output is to be in a continuous fanfold. This subparameter can also be coded as N.

Defaults

If you do not code a BURST parameter and the sysout data set is printed on a 3800 that has a burster-trimmer-stacker, JES uses an installation default specified at initialization.

Overrides

A BURST parameter on the sysout DD statement overrides the OUTPUT JCL BURST parameter.

OUTPUT JCL: BURST

Example of the BURST Parameter

```
//OUTDS1 OUTPUT BURST=YES
```

In this example, the output from the 3800 will be burst into separate sheets.

CHARS Parameter

Parameter Type: Keyword, optional

Purpose: Use the CHARS parameter to specify the name of one or more character-arrangement tables for printing the sysout data set on a 3800 Printing Subsystem.

Note:

- CHARS applies only for a data set printed on a 3800.
- STD applies only on a JES3 system.

References: For more information on character-arrangement tables, see the *3800 Printing Subsystem Programmer's Guide*. Refer to *Installation: System Generation* for information on how to choose during system generation particular groups, other than the Basic group, which is always available.

Syntax:

```
        ┌table-name                          ┐
CHARS=  │(table-name[,table-name]...)        │
        │STD                                 │
        │DUMP                                │
        └(DUMP[,table-name]...)              ┘
```

- You can omit the parentheses if you code only one table-name.
- Null positions in the CHARS parameter are invalid. For example, you **cannot** code CHARS=(,table-name) or CHARS=(table-name,,table-name).

Subparameter Definition

table-name
Names a character-arrangement table. Each table-name is 1 through 4 alphanumeric or national ($, #, @) characters. Code one to four names.

STD
Specifies the standard character-arrangement table. JES3 uses the standard table specified at initialization.

Note: STD is supported only on JES3 systems.

OUTPUT JCL: CHARS

DUMP
Requests a high-density dump of 204-character print lines from a 3800. If more than one table-name is coded, DUMP must be first.

Note: DUMP is valid only on the OUTPUT JCL statement referenced in a SYSABEND or SYSUDUMP DD statement that specifies a sysout data set for the dump.

Defaults

If you do not code the OUTPUT JCL CHARS parameter, JES uses the following, in order:

1. The DD CHARS parameter.
2. The DD UCS parameter value, if coded.
3. The OUTPUT JCL UCS parameter value, if coded.

If no character-arrangement table is specified on the DD or OUTPUT JCL statements, JES uses an installation default specified at initialization.

Overrides

A CHARS parameter on the sysout DD statement overrides the OUTPUT JCL CHARS parameter.

For a data set scheduled to the Print Services Facility (PSF), the PSF uses the following parameters, in override order, to select the font list.

1. Font list in the library member specified by an OUTPUT JCL PAGEDEF parameter.
2. DD CHARS parameter.
3. OUTPUT JCL CHARS parameter.
4. DD UCS parameter.
5. OUTPUT JCL UCS parameter.
6. JES installation default for the device.
7. Font list on the PAGEDEF parameter in the PSF cataloged procedure.

See "PAGEDEF Parameter" on page 17-51 for more information.

Requesting a High-Density Dump

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.

Example of the CHARS Parameter

```
//OUTDS2 OUTPUT CHARS=(GT12,GB12,GI12)
```

In this example, the output from the 3800 will be printed in three upper and lower case fonts: GT12, Gothic 12-pitch; GB12, Gothic Bold 12-pitch; and GI12, Gothic Italic 12-pitch.

CKPTLINE Parameter

Parameter Type: Keyword, optional

Purpose: Use the CKPTLINE parameter to specify the maximum number of lines in a logical page. JES uses this value, with the CKPTPAGE parameter, to determine when to take checkpoints while printing the sysout data set or transmitting the systems network architecture (SNA) data set.

Note: In a JES3 system, this parameter is supported when the Print Services Facility (PSF) prints the sysout data set on a 3800 Printing Subsystem Models 3, 6 and 8.

Syntax:

```
CKPTLINE=nnnnn
```

Subparameter Definition

nnnnn
Specifies the maximum number of lines in a logical page. nnnnn is a number from 0 through 32767.

Defaults

If you do not code the CKPTLINE parameter, JES2 uses an installation default specified at initialization. JES3 provides no installation default.

Example of the CKPTLINE Parameter

```
//OUTDS3 OUTPUT CKPTLINE=4000,CKPTPAGE=5
```

In this example, the sysout data set will be checkpointed after every 5 logical pages. Each logical page contains 4000 lines.

CKPTPAGE Parameter

Parameter Type: Keyword, optional

Purpose: Use the CKPTPAGE parameter to specify the number of logical pages:

- To be printed before JES takes a checkpoint.
- To be transmitted as a single systems network architecture (SNA) chain to an SNA work station before JES takes a checkpoint.

The number of lines in these logical pages is specified in the CKPTLINE parameter.

Note: In a JES3 system, this parameter is supported when the Print Services Facility (PSF) prints the sysout data on a 3800 Printing Subsystem Models 3, 6 and 8.

Syntax:

```
CKPTPAGE=nnnnn
```

Subparameter Definition

nnnnn
Specifies the number of logical pages to be printed or transmitted before the next sysout data set checkpoint is taken. nnnnn is a number from 1 through 32767.

Defaults

If you do not code the CKPTPAGE parameter, JES2 uses an installation default specified at initialization; the default may also indicate whether checkpoints are to be based on page count or time. JES3 provides no installation default.

Relationship to Other Parameters

If you code both the CKPTPAGE and CKPTSEC parameters:

- JES2 uses the value on the CKPTSEC parameter, provided the installation did not specify at initialization that checkpoints are to be based only on page count or time.
- JES3 uses the value on the CKPTPAGE parameter.

Example of the CKPTPAGE Parameter

```
//OUTDS4 OUTPUT CKPTPAGE=128,CKPTLINE=58
```

In this example, the sysout data set will be checkpointed after every 128 logical pages. Each logical page contains 58 lines.

CKPTSEC Parameter

Parameter Type: Keyword, optional

Purpose: Use the CKPTSEC parameter to specify how many seconds are to elapse between checkpoints of the sysout data set that JES is printing.

Note: In a JES3 system, this parameter is supported when the Print Services Facility (PSF) prints the sysout data set on a 3800 Printing Subsystem Models 3, 6 and 8.

Syntax:

```
CKPTSEC=nnnnn
```

Subparameter Definition

nnnnn
Specifies the number of seconds that is to elapse between checkpoints. nnnnn is a number from 1 through 32767.

Defaults

If you do not code the CKPTSEC parameter, JES2 uses an installation default specified at initialization; the default may also indicate whether checkpoints are to be based on page count or time. JES3 provides no installation default.

Relationship to Other Parameters

If you code both the CKPTPAGE and CKPTSEC parameters:

- JES2 uses the value on the CKPTSEC parameter, provided the installation did not specify at initialization that checkpoints are to be based only on page count or time.
- JES3 uses the value on the CKPTPAGE parameter.

Example of the CKPTSEC Parameter

```
//OUTDS5 OUTPUT CKPTSEC=120
```

In this example, the sysout data set will be checkpointed after every 120 seconds, or 2 minutes.

CLASS Parameter

Parameter Type: Keyword, optional

Purpose: Use the CLASS parameter to assign the sysout data set to an output class.

Note: If a sysout data set has the same class as the JOB statement MSGCLASS parameter, the job log appears on the same output listing as the sysout data set.

Syntax:

```
CLASS={class}
      {*    }
```

Subparameter Definition

class
Identifies the output class for the data set. The class is one character: A through Z or 0 through 9. The attributes of each output class are defined during JES initialization; specify the class with the desired attributes.

*****
Requests the output class in the MSGCLASS parameter on the JOB statement.

Overrides

The class subparameter of the DD statement SYSOUT parameter overrides the OUTPUT JCL CLASS parameter. On the DD statement, you must code a null class in order to use the OUTPUT JCL CLASS parameter; for example:

```
//OUTDS DD SYSOUT=(,),OUTPUT=*.OUT1
```

Held Classes in a JES2 System

An installation option at JES2 initialization determines if both the class for the sysout data set and the class for the job's messages must be held in order for a sysout data set to be held.

A sysout data set is held in the following cases:

- The sysout DD statement contains HOLD=YES.
- The sysout DD statement does not contain a HOLD parameter or contains HOLD=NO but requests a class that the installation defined as held and defined as:
  - Not requiring the message class to be a held class in order for the sysout data set to be held. The JOB statement MSGCLASS parameter can specify any class.
  - Requiring the message class to be a held class in order for the sysout data set to be held. The JOB MSGCLASS parameter must also specify a held class.

OUTPUT JCL: CLASS

A sysout data set is not held in the following cases:

- The sysout DD statement does not contain a HOLD parameter or contains HOLD=NO and requests:
  - A class that the installation defined as not held.
  - A class that the installation defined as held and defined as requiring the message class to be a held class in order for the sysout data set to be held. The JOB MSGCLASS parameter must specify a class that is not held.

Contact the installation to find out if holding the sysout class depends on a held MSGCLASS class.

Held Classes in a JES3 System

If CLASS specifies a class-name that is defined to JES3 as a held class for the output service hold queue (Q=HOLD), all of the new output characteristics might not be included in the data set on the writer queue when (1) the data set is moved from the hold queue to the output service writer queue (Q=WTR), (2) the data set includes an OUTPUT JCL statement, and (3) the NQ= or NCL= keyword is used.

For more information, see *MVS/XA SPL: JES3 Initialization and Tuning*.

Significance of Output Classes

To print this sysout data set and the messages from your job on the same output listing, code one of the following:

- The same output class in the DD SYSOUT parameter as in the JOB MSGCLASS parameter.
- DD SYSOUT=* to default to the JOB MSGCLASS output class.
- DD SYSOUT=(,) to default to one of the following:
  1. The CLASS parameter in an explicitly or implicitly referenced OUTPUT JCL statement. In this case, the OUTPUT JCL CLASS parameter should specify the same output class as the JOB MSGCLASS parameter.
  2. The JOB MSGCLASS output class, if no OUTPUT JCL statement is referenced or if the referenced OUTPUT JCL statement contains CLASS=*.

Examples of the CLASS Parameter

```
//OUTDS6 OUTPUT CLASS=D
//OUT1   DD     SYSOUT=(,),OUTPUT=*.OUTDS6
```

In this example, JES processes the sysout data set defined in DD statement OUT1 in output class D.

```
//PRINTALL JOB    ACCT123,MAEBIRD,MSGCLASS=H
//STEP1    EXEC   PGM=PRINTER
//OUTDS7   OUTPUT CLASS=*
//OUTPTR   DD     SYSOUT=(,),OUTPUT=*.OUTDS7
```

In this example, JES processes the sysout data set defined in DD statement OUTPTR in output class H, as specified in the JOB statement MSGCLASS parameter. The same result could be obtained by the following:

```
//PRINTALL JOB    ACCT123,MAEBIRD,MSGCLASS=H
//STEP1    EXEC   PGM=PRINTER
//OUTPTR   DD     SYSOUT=H
```

COMPACT Parameter

Parameter Type: Keyword, optional

Purpose: Use the COMPACT parameter to specify a compaction table for JES to use when sending the sysout data set, which is a systems network architecture (SNA) data set, to a SNA remote terminal.

Syntax:

```
COMPACT=compaction-table-name
```

Subparameter Definition

compaction-table-name
Specifies a compaction table by a symbolic name. The name is 1 through 8 alphanumeric characters. The symbolic name must be defined by the installation during JES initialization.

Defaults

If you do not code the COMPACT parameter, compaction is suppressed for the data set.

Overrides

This parameter overrides any compaction table value defined at the SNA remote terminal.

Example of the COMPACT Parameter

```
//OUTDS8 OUTPUT DEST=N555R222,COMPACT=TBL77
```

In this example, the sysout data set will be sent to remote terminal 222 at node 555; JES will use compaction table TBL77.

CONTROL Parameter

Parameter Type: Keyword, optional

Purpose: Use the CONTROL parameter to specify either that each logical record starts with a carriage control character or that the output is to be printed with single, double, or triple spacing.

Syntax:

```
          ┌PROGRAM┐
CONTROL=  │SINGLE │
          │DOUBLE │
          └TRIPLE ┘
```

Subparameter Definition

PROGRAM
Indicates that each logical record in the data set begins with a carriage control character. The carriage control characters are given in *Data Administration Guide*.

SINGLE
Indicates forced single spacing.

DOUBLE
Indicates forced double spacing.

TRIPLE
Indicates forced triple spacing.

Defaults

If you do not code the CONTROL parameter, JES3 uses an installation default specified at initialization.

In a JES2 system, an installation default can be provided for each local device by an operator command.

Example of the CONTROL Parameter

```
//OUTDS9 OUTPUT CONTROL=PROGRAM
```

In this example, the sysout data set is printed using the first character of each logical record for carriage control.

OUTPUT JCL: COPIES

COPIES Parameter

Parameter Type: Keyword, optional

Purpose: Use the COPIES parameter to specify how many copies of the sysout data set are to be printed. The printed output is in page sequence for each copy.

For printing on a 3800 Printing Subsystem, this parameter can instead specify how many copies of each page are to be printed before the next page is printed.

Syntax:

```
COPIES={nnn                                    }
       {(,(group-value[,group-value]...))      }
```

- You can omit the parentheses if you code only COPIES=nnn.
- The following are **not** valid:
  - A null group-value, for example, COPIES=(5,(,)) or COPIES=(5,)
  - A zero group-value, for example, COPIES=(5,(1,0,4))
  - A null within a list of group-values, for example, COPIES=(5,(1,,4))

Subparameter Definition

nnn

Specifies how many copies of the sysout data set are to be printed; each copy will be in page sequence order. nnn is 1 through 3 decimal numbers from 1 through 255 in a JES2 system and from 0 through 255 in a JES3 system.

For a data set printed on a 3800, JES ignores nnn if any group values are specified.

group-value

Specifies how many copies of each page are to be printed before the next page is printed. Each group-value is 1 through 3 decimal numbers from 1 through 255 in a JES2 system and from 1 through 254 in a JES3 system. You can code a maximum of eight group-values. Their sum must not exceed 255 or 254. The total copies of each page equals the sum of the group-values.

Note:

- This subparameter is valid only for 3800 output.
- For 3800 output, this subparameter overrides an nnn subparameter, if coded.

Defaults

For JES2, on the DD, OUTPUT JCL, or /*OUTPUT statement: if you do not code a COPIES parameter, code it incorrectly, or code COPIES=0, the system uses a default of 1, which is the default for the DD COPIES parameter.

For JES3, on the DD, OUTPUT JCL, or //*FORMAT statement: if you do not code a COPIES parameter, code it incorrectly, or code COPIES=0 on the DD statement, the system uses a default of 1, which is the default for the DD COPIES parameter.

Overrides

A COPIES parameter on the sysout DD statement overrides the OUTPUT JCL COPIES parameter.

If the OUTPUT JCL statement contains a FORMDEF parameter, which specifies a library member, the COPYGROUP parameter on a FORMDEF statement in that member overrides any group-value subparameters on the OUTPUT JCL COPIES parameter or the sysout DD COPIES parameter. For more information, see "FORMDEF Parameter" on page 17-38.

Relationship to Other Parameters

If the OUTPUT JCL or the sysout DD statement contains a FLASH parameter, JES prints with the forms overlay the number of copies specified in one of the following:

- COPIES=nnn, if the FLASH count is larger than nnn. For example, if COPIES=10 and FLASH=(LTHD,12) JES prints 10 copies, all with the forms overlay.
- The sum of the group-values specified in the COPIES parameter, if the FLASH count is larger than the sum. For example, if COPIES=(,(2,3,4) and FLASH=(LTHD,12) JES prints nine copies in groups, all with the forms overlay.
- The count subparameter in the FLASH parameter, if the FLASH count is smaller than nnn or the sum from the COPIES parameter. For example, if COPIES=10 and FLASH=(LTHD,7) JES prints seven copies with the forms overlay and three copies without.

Relationship to Other Control Statements

For JES2, if you request copies of the entire job on the JES2 /*JOBPARM COPIES parameter and also copies of the data set on the DD COPIES or OUTPUT JCL COPIES parameter, JES2 prints the number of copies equal to the **product** of the two requests.

OUTPUT JCL: COPIES

Examples of the COPIES Parameter

```
//RPTDS OUTPUT COPIES=4,FORMS=WKREPORT
```

This example asks JES to print four copies of the weekly report on forms named WKREPORT.

```
//EXPLD OUTPUT COPIES=(,(3)),FORMS=ACCT
```

This example asks JES to print the first page three times, then the second page three times, the third page three times, etc., on forms named ACCT.

```
//QUEST OUTPUT COPIES=(,(8,25,18,80)),FORMS=ANS
```

This example asks JES to print each page eight times before printing the next page, then 25 times before the next, then 18 times before the next, and finally 80 times before the next. The forms are named ANS.

```
//EXMP  OUTPUT COPIES=(5,(3,2))
```

This example asks JES to do one of the following:

- If the data set is printed on other than a 3800, to print five copies.
- If it is printed on a 3800, to print each page three times before printing the next page and then to print each page twice before printing the next page.

DATACK Parameter

Parameter Type: Keyword, optional

Purpose: Use the DATACK parameter to indicate whether or not print-positioning and invalid-character data-check errors are to be blocked or unblocked for printers accessed through the functional subsystem Print Services Facility (PSF).

A print-positioning error occurs when the designated position of any kind of printable information is beyond the limits of either the physical page, or the overlay or logical page of which it is part.

An invalid-character data-check error occurs when the hexadecimal representation of a text character has no mapping in the code page to a member of the font raster patterns.

If an error type is unblocked, the printer reports the error at the end of the page in which it occurs, and PSF processes the error and generates an error message. (See the PIMSG parameter for more information on the printing of error messages.)

If an error type is blocked, the printer does not report the error to PSF. Printing continues but data may be lost on the output.

References: For more information on data-check errors and their processing through PSF, see *Print Services Facility User's Programming Guide for MVS* or *Print Services Facility System Programmer's Guide for MVS*.

Syntax:

```
DATACK= { BLOCK   }
        { UNBLOCK }
        { BLKCHAR }
        { BLKPOS  }
```

Subparameter Definition

BLOCK
Indicates that print-positioning errors and invalid-character errors are not reported to PSF.

UNBLOCK
Indicates that print-positioning errors and invalid-character errors are reported to PSF.

BLKCHAR
Indicates that invalid-character errors are blocked, and not reported to PSF. Print-positioning errors are reported normally.

BLKPOS
Indicates that print-positioning errors are blocked, and not reported to PSF. Invalid-character errors are reported normally.

OUTPUT JCL: DATACK

Defaults

If you do not code the DATACK parameter, the DATACK specification from the PSF PRINTDEV statement is used. If not specified in the PRINTDEV statement, the default is BLOCK.

Relationship to Other Parameters

If DATACK is specified as UNBLOCK, BLKCHAR, or BLKPOS, and an unblocked error occurs, the printer reports the error to PSF which processes the error. The coding of the PIMSG parameter then determines whether or not printing of the data set continues after the page in error, and if error messages are printed at the end of the data set.

Example of the DATACK Parameter

```
//OUTDS1 OUTPUT DATACK=BLKCHAR,PIMSG=(YES,0)
```

In this example, when a print-position error occurs, it is reported to the user via a printed error message. If an invalid-character error occurs, it is not reported. In either case, the printing of the data set continues, and all functional subsystem messages are printed.

DEFAULT Parameter

Parameter Type: Keyword, optional

Purpose: Use the DEFAULT parameter to specify that this OUTPUT JCL statement can or cannot be implicitly referenced by a sysout DD statement. An OUTPUT JCL statement that contains a DEFAULT=YES parameter is called a default OUTPUT JCL statement.

Syntax:

```
DEFAULT= (YES)
         (Y  )
         (NO )
         (N  )
```

Subparameter Definition

YES
Indicates that this OUTPUT JCL statement can be implicitly referenced by sysout DD statements. This subparameter can also be coded as Y.

NO
Indicates that this OUTPUT JCL statement cannot be implicitly referenced by sysout DD statements. This subparameter can also be coded as N.

Defaults

If you do not code DEFAULT=YES, the default is NO. In order to take effect, an OUTPUT JCL statement without DEFAULT=YES must be explicitly referenced in an OUTPUT parameter on a sysout DD statement.

Location in the JCL

- A step-level OUTPUT JCL statement appears within a step, that is, anywhere after the first EXEC statement in a job.
- A job-level OUTPUT JCL statement appears after the JOB statement and before the first EXEC statement.
- You can place more than one job- or step-level default OUTPUT JCL statement in a job or step.
- You must place an OUTPUT JCL statement in the input stream **before** any sysout DD statement that explicitly or implicitly refers to it.

OUTPUT JCL: DEFAULT

References to Default OUTPUT JCL Statements

- A sysout DD statement makes an explicit reference in an OUTPUT parameter that specifies the name of an OUTPUT JCL statement.
- A sysout DD statement makes an implicit reference when it does not contain an OUTPUT parameter, and the job or step contains one or more default OUTPUT JCL statements.
- A sysout DD statement implicitly references all step-level default OUTPUT JCL statements in the same step.
- A sysout DD statement implicitly references all job-level default OUTPUT JCL statements when the step containing the DD statement does not contain any step-level default OUTPUT JCL statements.
- A sysout DD statement can explicitly reference a default OUTPUT JCL statement.

Example of the DEFAULT Parameter

```
//EXMP2     JOB      ACCT555,MAEBIRD,MSGCLASS=B
//OUTDAL    OUTPUT   DEFAULT=YES,DEST=DALLAS
//OUTPOK    OUTPUT   DEST=POK
//STEP1     EXEC     PGM=REPORT
//OUTHERE   OUTPUT   CLASS=D
//SYSIN     DD       *
             .
             .
             .
/*
//WKRPT     DD       UNIT=VIO,DISP=(,PASS)
//RPT1      DD       SYSOUT=(,),OUTPUT=*.OUTHERE
//RPT2      DD       SYSOUT=A
//STEP2     EXEC     PGM=SUMMARY
//OUTHQ     OUTPUT   DEFAULT=YES,DEST=HQ
//WKDATA    DD       UNIT=VIO,DISP=(OLD,DELETE),DSNAME=*.STEP1.WKRPT
//MONTH     DD       SYSOUT=(,),OUTPUT=*.STEP1.OUTHERE
//SUM       DD       SYSOUT=A
//FULRPT    DD       SYSOUT=A,OUTPUT=(*.OUTDAL,*.OUTPOK)
```

In this example, the JOB named EXMP2 contains two job-level OUTPUT JCL statements: OUTDAL and OUTPOK. OUTDAL is a default OUTPUT JCL statement because it contains DEFAULT=YES; OUTDAL can be implicitly referenced by a sysout DD statement. OUTPOK must be explicitly referenced in a sysout DD OUTPUT parameter for its processing options to be used. The purpose of both of these OUTPUT JCL statements is to specify a destination for a sysout data set.

STEP1 contains a step-level OUTPUT JCL statement: OUTHERE. The purpose of this statement is to specify that JES process the data set locally in output class D. OUTHERE can be used only if it is explicitly referenced.

STEP2 contains a step-level default OUTPUT JCL statement: OUTHQ. The purpose of this statement is to specify a destination for a sysout data set. OUTHQ can be implicitly referenced.

The references in this job are as follows:

- In STEP1 and STEP2, sysout DD statements RPT1 and MONTH explicitly reference OUTPUT JCL statement OUTHERE. These two sysout data sets are printed locally in the same output class.

  Note: You can explicitly reference an OUTPUT JCL statement in a preceding job step.

- In STEP1, DD statement RPT2 implicitly references OUTPUT JCL statement OUTDAL. This implicit reference occurs because all of the following are true:

  1. DD statement RPT2 contains a SYSOUT parameter but does not contain an OUTPUT parameter. Thus, this DD statement is making an implicit reference.

  2. STEP1 does not contain a default OUTPUT JCL statement, so the implicit reference must be to job-level default OUTPUT JCL statements.

  3. OUTDAL is the only job-level default OUTPUT JCL statement.

- In STEP2, DD statement SUM implicitly references OUTPUT JCL OUTHQ because all of the following are true:

  1. DD statement SUM contains a SYSOUT parameter but does not contain an OUTPUT parameter. Thus, this DD statement is making an implicit reference.

  2. STEP2 contains a default OUTPUT JCL statement: OUTHQ. Therefore, the implicit reference is to OUTHQ and cannot be to any job-level default OUTPUT JCL statements.

- In STEP2, DD statement FULRPT explicitly references OUTPUT JCL statements OUTDAL and OUTPOK.

DEST Parameter

Parameter Type: Keyword, optional

Purpose: Use the DEST parameter to specify a destination for the sysout data set. The DEST parameter can send a sysout data set to a remote or local terminal, a node, a node and remote work station, a local device or group of devices, or a node and userid.

Syntax:

```
DEST=destination

The destination subparameter for JES2 is one of the following:

     LOCAL
     name
     Nnnnn
     NnnRmmmm
     NnnnRmmm
     NnnnnRmm
     nodename.userid
     Rnnnn
     RMnnnn
     RMTnnnn
     Unnnn

The destination subparameter for JES3 is one of the following:

     ANYLOCAL
     device-name
     group-name
     nodename
     nodename.userid
```

Subparameter Definition for JES2 Systems

LOCAL
Indicates any local device.

name
Identifies a local or remote device by a symbolic name defined by the installation during JES2 initialization. The name is 1 through 8 alphanumeric or national ($, #, @) characters.

Nnnnn
Identifies a node. nnnn is 1 through 4 decimal numbers from 1 through 1000. For example, N103.

NnnRmmmm
NnnnRmmm
NnnnnRmm
Identifies a node and a remote work station connected to the node. The node number, indicated in the format by n, is 1 through 4 decimal numbers from 1 through 1000. The remote work station number, indicated in the format by m, is 1 through 4 decimal

numbers from 1 through 9999. Do not code leading zeros in n or m. The maximum number of digits for n and m combined cannot exceed six.

Note: R0 is equivalent to LOCAL specified at node Nn.

nodename.userid
Identifies a destination node and a VM or a TSO userid, a remote workstation, or a symbolic name defined at the destination node. The nodename is a symbolic name defined at the node of execution. nodename is 1 through 8 alphanumeric or national ($, #, @) characters. userid is 1 through 8 alphanumeric or national ($, #, @) characters, and must be defined at the specified node.

Note: If a data set is queued for transmission and an operator changes its destination, the userid portion of the original routing is lost.

Rnnnn
RMnnnn
RMTnnnn
Identifies a remote terminal. nnnn is 1 through 4 decimal numbers from 1 through 9999. Note that with remote pooling, the installation may translate this route code to another route code.

Note: R0 is equivalent to LOCAL.

Unnnn
Identifies a local terminal with special routing. nnnn is 1 through 4 decimal numbers from 1 through 9999.

Subparameter Definition for JES3 Systems

ANYLOCAL
Indicates any local device.

device-name
Identifies a local device by a symbolic name defined by the installation during JES3 initialization. device-name is 1 through 8 alphanumeric or national ($, #, @) characters.

group-name
Identifies a group of local devices, an individual remote station, or a group of remote stations by a symbolic name defined by the installation during JES3 initialization. group-name is 1 through 8 alphanumeric or national ($, #, @) characters.

nodename
Identifies a node by a symbolic name defined by the installation during JES3 initialization. The node is 1 through 8 alphanumeric or national ($, #, @) characters. If the node you specify is the same as the node you are working on, JES3 treats the output as though you had specified ANYLOCAL.

nodename.userid
Identifies a destination node and a VM or TSO userid at that node. The nodename is a symbolic name defined by the installation during initialization. The userid must be defined at the node. A userid requires a node; therefore, code DEST=nodename.userid. Do not code a userid without a node.

OUTPUT JCL: DEST

Defaults

If you do not code a DEST parameter, JES directs the sysout data set to the default destination for the input device from which the job was submitted.

If a specified destination is invalid, the job fails.

Overrides

A DEST parameter on the sysout DD statement overrides the OUTPUT JCL DEST parameter.

Relationship to Other Parameters

For JES3, you can code the DEST=nodename parameter with the OUTPUT JCL WRITER=name parameter; however, do not code DEST=nodename.userid with WRITER=name.

Examples of the DEST Parameter

```
//REMOT1 OUTPUT DEST=R444
```

In this example, JES2 sends the sysout data set to remote terminal 444.

```
//REMOT2 OUTPUT DEST=STAT444
```

In this example, JES sends the sysout data set to an individual remote station named by the installation STAT444.

```
//REMOT3 OUTPUT DEST=KOKVMBB8.DP58HHHD
```

In this example, JES sends the sysout data set to VM userid DP58HHHD at node KOKVMBB8.

FCB Parameter

Parameter Type: Keyword, optional

Purpose: Use the FCB parameter to specify:

- The forms control buffer (FCB) image JES is to use to guide printing of the sysout data set by a 1403 Printer, 3211 Printer, 3203 Printer Model 5, 3800 Printing Subsystem, 4245 Printer, or 4248 Printer, or by a printer supported by systems network architecture (SNA) remote job entry (RJE).

- The carriage control tape JES is to use to control printing of the sysout data set by a 1403 Printer or by a printer supported by SNA RJE.

- The data-protection image JES is to use to control output by a 3525 Card Punch.

The FCB image specifies how many lines are to be printed per inch and the length of the form. JES loads the image into the printer's forms control buffer. The FCB image is stored in SYS1.IMAGELIB. IBM provides three standard FCB images:

- STD1, which specifies 6 lines per inch on an 8.5-inch-long form. (3211 and 3203-5 only)
- STD2, which specifies 6 lines per inch on an 11-inch-long form. (3211 and 3203-5 only)
- STD3, which in a JES3 system specifies 8 lines per inch for a dump. (3800 only)

References: For more information on the forms control buffer, see:

MVS/XA System-Data Administration,
Programming Support for the IBM 3505 Card Reader and IBM 3525 Card Punch, or
3800 Printing Subsystem Programmer's Guide.

Syntax:

```
FCB={fcb-name}
    {STD     }
```

- Code the fcb-name as STD1 or STD2 only to request the IBM-supplied images.
- Code the fcb-name as STD3 only for a high-density dump in a JES3 system.

Subparameter Definition

fcb-name
Identifies the FCB image. The name is 1 through 4 alphanumeric or national ($, #, @) characters and is the last characters of a SYS1.IMAGELIB member name:

- FCB2xxxx member, for a 3211, a 3203 Model 5, or a printer supported by SNA.
- FCB3xxxx member, for a 3800.
- FCB4xxxx member, for a 4248.

STD
Indicates the standard FCB. JES3 uses the standard FCB specified at JES3 initialization.

Note: STD is supported only on JES3 systems.

Defaults

If you do not code the FCB parameter, the system checks the FCB image in the printer's forms control buffer; if it is a default image, as indicated by its first byte, JES uses it. If it is not a default image, JES loads the FCB image that is the installation default specified at JES initialization.

Overrides

An FCB parameter on the sysout DD statement overrides the OUTPUT JCL FCB parameter.

Relationship to Other Parameters

The FCB parameter is mutually exclusive with the FRID subparameter of the DD statement DCB parameter.

Requesting a High-Density Dump

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.

Example of the FCB Parameter

```
//OUTDS1 OUTPUT FCB=AA33
```

In this example, JES will print the sysout data set using the FCB image named AA33.

FLASH Parameter

Parameter Type: Keyword, optional

Purpose: Use the FLASH parameter to identify the forms overlay to be used in printing the sysout data set on a 3800 Printing Subsystem and, optionally, to specify the number of copies on which the forms overlay is to be printed.

Note: FLASH applies only for a data set printed on a 3800.

References: For information on forms overlays, see the *Forms Design Reference Guide for the IBM 3800 Printing Subsystem.*

Syntax:

```
        ⎧overlay-name          ⎫
FLASH=  ⎪(overlay-name[,count]) ⎪
        ⎨(,count)              ⎬
        ⎪NONE                  ⎪
        ⎩STD                   ⎭
```

The count subparameter is optional. If you omit it, you can omit the parentheses.

Subparameter Definition

overlay-name
Identifies the forms overlay frame that the operator is to insert into the printer before printing begins. The name is 1 through 4 alphanumeric or national ($, #, @) characters.

count
Specifies the number, 0 through 255, of copies that JES is to flash with the overlay, beginning with the first copy printed. Code a count of 0 to flash **no** copies.

NONE
Suppresses flashing for this sysout data set.

If FLASH=NONE is on an OUTPUT JCL statement in a job to be executed at a remote node, JES3 sets the overlay-name to zero before sending the job to the node.

STD
Indicates the standard forms flash overlay. JES3 uses the standard forms overlay specified at JES3 initialization.

Note: STD is supported only on JES3 systems.

OUTPUT JCL: FLASH

Defaults

If you do not code a FLASH parameter or an installation default was not specified at JES2 or JES3 initialization, forms are not flashed.

If you specify an overlay-name without specifying a count, all copies are flashed. That is, the default for count is 255.

Overrides

A FLASH parameter on the sysout DD statement overrides the OUTPUT JCL FLASH parameter.

Relationship to Other Parameters

If the OUTPUT JCL or the sysout DD statement also contains a COPIES parameter, JES prints with the forms overlay the number of copies specified in one of the following:

- COPIES=nnn, if the FLASH count is larger than nnn. For example, if COPIES=10 and FLASH=(LTHD,12) JES prints 10 copies, all with the forms overlay.
- The sum of the group-values specified in the COPIES parameter, if the FLASH count is larger than the sum. For example, if COPIES=(,(2,3,4)) and FLASH=(LTHD,12) JES prints nine copies in groups, all with the forms overlay.
- The count subparameter in the FLASH parameter, if the FLASH count is smaller than nnn or the sum from the COPIES parameter. For example, if COPIES=10 and FLASH=(LTHD,7) JES prints seven copies with the forms overlay and three copies without.

Verification of Forms Overlay Frame

Before printing starts, the system requests the operator to load the specified forms overlay frame in the printer. A frame must be loaded but the system cannot verify that it is the correct frame.

Printing without Flashing

To print without flashing, specify one of the following:

- FLASH=NONE on the DD or OUTPUT JCL statement.
- Omit the FLASH parameter on all of the statements for the data set and on all JES initialization statements.
- FLASH=(,0) on the OUTPUT JCL statement.

Example of the FLASH Parameter

```
//OUTDS1 OUTPUT COPIES=16,FLASH=(LTHD,7)
```

In this example, JES issues a message to the operator requesting that the forms overlay frame named LTHD be inserted in the printer. Then JES prints the first seven copies of the sysout data set with the forms overlay and the last nine without.

FORMDEF Parameter

Parameter Type: Keyword, optional

Purpose: Use the FORMDEF parameter to identify a library member that contains statements to tell the Print Services Facility (PSF) how to print the sysout data set on a page-mode printer (such as the 3800 Printing Subsystem Model 3). The statements can specify the following:

- Overlay forms to be used during printing.
- Location on the page where overlays are to be placed.
- Suppressions that can be activated for specified page formats.

The member must be in the library named in the cataloged procedure that was used to initialize the PSF.

Note: FORMDEF applies only for data sets printed on a page-mode printer (such as the 3800 Model 3).

References: For more information, see *Print Management Facility User's Guide and Reference* and *3800 Printing Subsystem Programmer's Guide.*

Syntax:

```
FORMDEF=membername
```

Subparameter Definition

membername
Specifies the name of a library member. membername is 1 through 6 alphanumeric or national ($, #, @) characters; the first two characters are pre-defined by the system.

Overrides

The library member specified by the OUTPUT JCL FORMDEF parameter can contain:

- Statements that override the installation's FORMDEF defaults in the PSF cataloged procedure.
- A FORMDEF statement with a COPYGROUP parameter. The COPYGROUP parameter overrides any group-value subparameters on the OUTPUT JCL COPIES parameter or the sysout DD COPIES parameter.

  Note: The FORMDEF statement in the library member does not override a sysout DD or OUTPUT JCL COPIES=nnn parameter.

Example of the FORMDEF Parameter

```
//PRINT3 OUTPUT FORMDEF=JJPRT
```

In this example, PSF is to print the sysout data set on a 3800 Model 3 according to the parameters in the library member JJPRT.

FORMS Parameter

Parameter Type: Keyword, optional

Purpose: Use the FORMS parameter to identify the forms on which the sysout data set is to be printed or punched.

Syntax:

```
FORMS={form-name}
      {STD      }
```

Subparameter Definition

form-name
Identifies the print or punch forms. form-name is 1 through 8 alphanumeric or national ($, #, @) characters.

STD
Indicates that JES3 is to use the standard form specified at JES3 initialization.

Note: STD is supported only on JES3 systems.

Defaults

If you do not code a form-name subparameter, JES uses an installation default specified at initialization.

Overrides

The form-name subparameter of the SYSOUT parameter on the sysout DD statement overrides the OUTPUT JCL FORMS parameter. Note that the SYSOUT form-name subparameter can be only four characters maximum while both the OUTPUT JCL FORMS form-name and the JES initialization default form names can be eight characters maximum.

Example of the FORMS Parameter

```
//OUTDS1 OUTPUT FORMS=ACCT4010
```

In this example, the sysout data set will be printed on forms named ACCT4010.

GROUPID Parameter

Parameter Type: Keyword, optional, JES2 only

Purpose: Use the GROUPID parameter to specify that the sysout data set belongs to an output group. The data sets in an output group are processed together in the same location and time. Data sets to be grouped should have similar characteristics: the same output class, destination, process mode, and external writer name.

Note: GROUPID is supported only on JES2 systems.

Syntax:

```
GROUPID=output-group
```

Subparameter Definition

output-group
Specifies the name of an output group. The output-group is 1 through 8 alphanumeric characters and is selected by the programmer to define an output group for this job. The name is not installation-defined.

Relationship to Other Control Statements

If you code FREE=CLOSE on a sysout DD statement that references an OUTPUT JCL statement containing a GROUPID parameter, JES2 will not group the data sets into one output group. Instead, JES2 produces one copy of the sysout data set for each OUTPUT JCL statement that the DD statement references.

OUTPUT JCL: GROUPID

Examples of the GROUPID Parameter

```
//EXMP5   JOB      ACCT1984,MAEBIRD,MSGCLASS=A
//OUTRPT  OUTPUT   GROUPID=RPTGP,DEFAULT=YES,DEST=TDC
//STEP1   EXEC     PGM=RPTWRIT
//SYSIN   DD       *
          .
          .
          .
/*
//RPTDLY  DD       SYSOUT=C
//RPTWK   DD       SYSOUT=C
```

In this example, the DD statements RPTDLY and RPTWK implicitly reference the default OUTPUT JCL statement OUTRPT. JES2 creates two output groups:

1. Group RPTGP is created because of the GROUPID parameter in the OUTPUT JCL statement. It contains the two reports from the sysout DD statements RPTDLY and RPTWK and is printed at the destination TDC. The programmer named this group RPTGP.

2. The other group is named by JES2. It contains the system-managed data set for the job's messages.

```
//EXAMP     JOB      MSGCLASS=A
//JOBOUT    OUTPUT   GROUPID=SUMM,DEST=HQS,CHARS=GT10
//STEP1     EXEC     PGM=RWRITE
//OUT1      OUTPUT   FORMS=STD,CHARS=GS10,DEST=LOCAL
//RPT1      DD       SYSOUT=A,OUTPUT=(*.OUT1,*.JOBOUT)
//STEP2     EXEC     PGM=SWRITE
//OUT2      OUTPUT   FORMS=111,CHARS=GB10,DEST=LOCAL
//RPT2      DD       SYSOUT=B,OUTPUT=(*.OUT2,*.JOBOUT)
```

This job causes JES2 to produce five sets of output:

1.1.1, containing the system-managed data sets. This set is specified through the JOB statement MSGCLASS parameter.

SUMM.1.1, containing a copy of the data set defined by DD statement RPT1. This set is specified through the second OUTPUT subparameter: *.JOBOUT. It is for output class A.

SUMM.2.1, containing a copy of the data set defined by DD statement RPT2. This set is specified through the second OUTPUT subparameter: *.JOBOUT. Because it is for output class B, it is in a separate subgroup from the SUMM.1.1 subgroup.

4.1.1, containing a copy of the data set defined by DD statement RPT1. This set is specified through the first OUTPUT subparameter: *.OUT1.

5.1.1, containing a copy of the data set defined by DD statement RPT2. This set is specified through the first OUTPUT subparameter: *.OUT2.

INDEX Parameter

Parameter Type: Keyword, optional, JES2 only

Purpose: Use the INDEX parameter to set the left margin for output on a 3211 Printer with the indexing feature. The width of the print line is reduced by the INDEX parameter value.

Note: INDEX is supported only on JES2 systems and only for output printed on a 3211 with the indexing feature. JES2 ignores the INDEX parameter if the printer is not a 3211 with the indexing feature.

Syntax:

```
INDEX=nn
```

Subparameter Definition

nn

Specifies how many print positions the left margin on the 3211 output is to be indented. nn is a decimal number from 1 through 31. n=1 indicates flush-left; n=2 through n=31 indent the print line by n-1 positions.

Defaults

The default is 1, which indicates flush left. Thus, if you do not code an INDEX or LINDEX parameter, JES2 prints full-width lines.

Relationship to Other Parameters

INDEX and LINDEX are mutually exclusive; if you code both, JES2 uses the last one encountered. Note that you cannot index both the left and right margins.

Example of the INDEX Parameter

```
//OUT17  OUTPUT INDEX=6
```

In this example, because the printed report is to be stapled, extra space is needed on the left. Assuming the data set is printed on a 3211 with the indexing feature, all lines are indented 5 print positions from the left page margin.

JESDS Parameter

Parameter Type: Keyword, optional

Purpose: Use the JESDS parameter to process the job's system-managed data sets according to the parameters on this OUTPUT JCL statement. The system-managed data sets consist of:

- The job log, which is a record of job-related information for the programmer. Printing of the job log is controlled by two JOB statement parameters: the MSGLEVEL parameter controls what is printed and the MSGCLASS parameter controls the system output class.
- The job's hard-copy log, which is a record of all message traffic for the job to and from the operator console.
- System messages for the job.

References: For more information on the job log, see *System Commands*.

Syntax:

```
JESDS=(ALL)
      {JCL}
      {LOG}
      (MSG)
```

Subparameter Definition

ALL
Indicates all of the job's system-managed data sets.

JCL
Indicates all job control statements in the input stream, that is, all JCL statements and JES2 or JES3 statements, plus all procedure statements from any in-stream or cataloged procedure a job step calls, plus all messages about job control statements.

LOG
Indicates the job's hard-copy log, which contains the JES and operator messages about the job's processing: allocation of devices and volumes, execution and termination of job steps and the job, and disposition of data sets.

MSG
Indicates any system messages for this job.

Overrides

The NOLOG parameter on a JES2 /*JOBPARM statement overrides the OUTPUT JCL JESDS=ALL parameter.

Location in the JCL

Place an OUTPUT JCL statement containing JESDS before the first EXEC statement of the job. An OUTPUT JCL statement containing JESDS placed after an EXEC statement is a JCL error.

You can place more than one OUTPUT JCL statement containing JESDS before the first EXEC statement. JES creates a copy of the job's system data sets for each.

Destination for the System Data Sets

If you want the job's system data sets processed at a particular destination, code a DEST parameter on the OUTPUT JCL statement containing JESDS. Otherwise, JES routes the system data sets to a local device.

JES2 Processing with JESDS

JES2 processes OUTPUT JCL statements for system-managed data sets (JESDS parameter) only if a job completes execution.

System-managed data sets are not processed for the following jobs because the jobs do not complete execution:

- Jobs that specify a TYPRUN value on the JOB statement that prevents execution, such as COPY or SCAN.
- Jobs that do not execute because of a JCL error, an error in a JES2 control statement, or a system failure in JES2 input processing.
- Jobs that reach execution but do not complete execution because an MVS system failure occurs that requires an IPL.

JES3 Processing with JESDS

System-managed data sets are not processed by JES3 for the following jobs because the jobs do not complete execution:

- Jobs that specify a TYPRUN value on the JOB statement that prevents execution, such as SCAN.
- Jobs that do not execute because of a JCL error.

OUTPUT JCL: JESDS

Example of the JESDS Parameter

```
//EXMP     JOB      MSGCLASS=A
//OUT1     OUTPUT   JESDS=ALL
//OUT2     OUTPUT   JESDS=ALL,DEST=AUSTIN
           .
           .
           .
```

In this example, JES produces two copies of the system-managed data sets: one copy for OUTPUT JCL statement OUT1 and one copy for OUTPUT JCL statement OUT2. The copy for statement OUT2 is sent to AUSTIN.

LINDEX Parameter

Parameter Type: Keyword, optional, JES2 only

Purpose: Use the LINDEX parameter to set the right margin for output on a 3211 Printer with the indexing feature. The width of the print line is reduced by the LINDEX parameter value.

Note: LINDEX is supported only on JES2 systems and only for output printed on a 3211 with the indexing feature. JES2 ignores the LINDEX parameter if the printer is not a 3211 with the indexing feature.

Syntax:

```
LINDEX=nn
```

Subparameter Definition

nn

Specifies how many print positions the right margin on the 3211 output is to be moved in from the full page width. nn is a decimal number from 1 through 31. n=1 indicates flush-right; n=2 through n=31 move the right margin over by n-1 positions.

Defaults

The default is 1, which indicates flush right. Thus, if you do not code an INDEX or LINDEX parameter, JES2 prints full-width lines.

Relationship to Other Parameters

INDEX and LINDEX are mutually exclusive; if you code both, JES2 uses the last one encountered. Note that you cannot index both the left and right margins.

Example of the LINDEX Parameter

```
//OUT18  OUTPUT LINDEX=21
```

In this example, the author of the report wanted extra space on the right side of the paper for notes. Assuming the data set is printed on a 3211 with the indexing feature, all lines are ended 20 print positions from the right page margin.

LINECT Parameter

Parameter Type: Keyword, optional, JES2 only

Purpose: Use the LINECT parameter to specify the maximum number of lines JES2 is to print on each output page.

Note: LINECT is supported only on JES2 systems.

Syntax:

```
LINECT=nnn
```

Subparameter Definition

nnn
Specifies the maximum number of lines JES2 is to print on each page. nnn is a number from 0 through 255.

Specify LINECT=0 to keep JES2 from starting a new page when the number of lines exceeds the JES2 initialization parameter.

Defaults

If you do not code the LINECT parameter, JES2 obtains the value from one of the following sources, in order:

1. The linect field of the accounting information parameter on the JOB statement.
2. The installation default specified at JES2 initialization.

Example of the LINECT Parameter

```
//PRNTDS OUTPUT LINECT=45
```

In this example, JES2 will start a new page after every 45 lines.

MODIFY Parameter

Parameter Type: Keyword, optional

Purpose: Use the MODIFY parameter to specify a copy-modification module that tells JES how to print the sysout data set on a 3800 Printing Subsystem. The module can specify the following:

- Legends.
- Column headings.
- Where and on which copies the data is to be printed.

The module is defined and stored in SYS1.IMAGELIB using the IEBIMAGE utility program.

Note: MODIFY applies only for the 3800 Printing Subsystem Model 1 and 2 and the 3800 Printing Subsystem Model 3, 6 and 8 in compatibility mode. For page-mode printers (such as the 3800 Model 3), use the FORMDEF and PAGEDEF parameters to obtain the same functions.

References: For more information on the copy modification module and the IEBIMAGE utility program, see *Data Administration: Utilities.*

Syntax:

```
MODIFY={module-name              }
       {([module-name][,trc])    }
```

- You can omit the module-name, thereby obtaining the initialization default. For example, MODIFY=(,2).
- The trc subparameter is optional. If you omit it, you can omit the parentheses.

Subparameter Definition

module-name
Identifies a copy-modification module in SYS1.IMAGELIB. The module-name is 1 through 4 alphanumeric or national ($, #, @) characters.

trc

Identifies which character-arrangement table named in the CHARS parameter is to be used. This **table reference character** is 0 for the first table-name specified, 1 for the second, 2 for the third, or 3 for the fourth. The CHARS parameter used is on the following, in override order:

1. The DD statement.
2. This OUTPUT JCL statement.
3. A statement in the library member specified on the OUTPUT JCL PAGEDEF parameter.
4. A statement in the SYS1.IMAGELIB member obtained by default.
5. A JES3 initialization statement.

Defaults

If you do not code module-name in the MODIFY parameter, JES3 uses an installation default specified at initialization. JES2 provides no installation default at initialization.

If you do not specify **trc**, the default is 0. If the trc value is greater than the number of table-names in the CHARS parameter, JES2 uses the first character-arrangement table named in the CHARS parameter and JES3 uses the last character-arrangement table named in the CHARS parameter.

Overrides

A MODIFY parameter on the sysout DD statement overrides the OUTPUT JCL MODIFY parameter.

Relationship to Other Parameters

The second character of each logical record can be a TRC code, so that each record can be printed in a different font. This way of specifying fonts is indicated by the OUTPUT JCL TRC parameter.

Example of the MODIFY Parameter

```
//OUTDS1 OUTPUT  CHARS=(GT12,GB12,GI12),MODIFY=(MODA,2)
```

In this example, JES loads the MODA module in SYS1.IMAGELIB into the 3800 and uses GI12, Gothic Italic 12-pitch font, which is the third table name specified in the CHARS parameter.

PAGEDEF Parameter

Parameter Type: Keyword, optional

Purpose: Use the PAGEDEF parameter to identify a library member that contains statements to tell the Print Services Facility (PSF) how to print the sysout data set on a page-mode printer (such as a 3800 Printing Subsystem Model 3). The statements can specify the following:

- Logical page length and width.
- Fonts.
- Page segments.
- Multiple page types or formats.
- Lines within a page; for example:
  - Line origin.
  - Carriage controls.
  - Spacing.
- Multiple logical pages on a physical page.

The member must be in the library named in the cataloged procedure that was used to initialize the PSF.

Note: PAGEDEF applies only for data sets printed on a page-mode printer (such as a 3800 Model 3).

References: For more information, see *Print Management Facility User's Guide and Reference* and *3800 Printing Subsystem Programmer's Guide*.

Syntax:

```
PAGEDEF=membername
```

Subparameter Definition

membername
Specifies the name of the library member. membername is 1 through 6 alphanumeric or national ($, #, @) characters; the first two characters are pre-defined by the system.

OUTPUT JCL: PAGEDEF

Overrides

The statements in the library member specified by the OUTPUT JCL PAGEDEF parameter override the installation's PAGEDEF defaults in the PSF cataloged procedure.

The PSF uses the following parameters, in override order, to select the font list:

1. Font list in the library member specified by an OUTPUT JCL PAGEDEF parameter.
2. DD CHARS parameter.
3. OUTPUT JCL CHARS parameter.
4. DD UCS parameter.
5. OUTPUT JCL UCS parameter.
6. JES installation default for the device.
7. Font list on the PAGEDEF parameter in the PSF cataloged procedure.

Example of the PAGEDEF Parameter

```
//OUTDS1 OUTPUT PRMODE=PAGE,PAGEDEF=SSPGE
```

In this example, PSF is to print the sysout data set on a 3800 Model 3 operating in page mode. The printing is to be done according to the parameters in the library member SSPGE.

PIMSG Parameter

Parameter Type: Keyword, optional

Purpose: Use the PIMSG parameter to indicate the handling of messages by the functional subsystem Print Services Facility (PSF). PIMSG is used to specify whether all error messages are to be printed, and the number of errors sufficient to cause the printing of the data set to be terminated.

The functional subsystem PSF accumulates messages in the PSF address space for errors that occur while processing line-mode and page-mode data sets.

When you code PIMSG=YES, the system prints all these messages at the end of the output data set.

When you code PIMSG=NO, no messages are printed unless there is an error that forces premature termination of the printing of the data set. If an error occurs, the system prints the set of messages (called a message group) associated with the error that caused the termination.

As errors are detected by PSF or reported to PSF by the printer, a count is kept of the associated message groups. When the count equals the number specified on the PIMSG parameter, PSF terminates the printing of the current data set. PSF interprets a count of zero as infinite, and does not terminate the printing of the data set on the basis of the number of errors detected.

Note: PIMSG can be specified only for data sets printed through PSF.

Syntax:

```
PIMSG= {(YES[,msg-count])}
       {(NO[,msg-count]) }
```

- You can omit the parentheses if you do not specify msg-count.

Subparameter Definition

YES
Requests that the system print all messages generated by the functional subsystem. This subparameter can also be coded as Y.

NO
Requests that the system print no error messages, except if the printing of the data set is prematurely terminated. If a terminating error occurs, only the set of messages (called a message group) associated with the error that caused the termination is printed. This subparameter can also be coded as N.

msg-count
Requests that the system cancel the printing of the current data set after the specified number of errors (as represented by the associated message groups) have been detected by PSF or reported to PSF by the printer. In this context, errors refers to data-stream errors, and errors resulting from any malfunction that would cause the printer to halt,

such as a mechanism failure, or out-of-paper condition. However, these errors do not include those caused by operator intervention.

Valid values for msg-count are 0-999, where 0 is interpreted as infinite.

The types of errors that increment the message count are those that induce a message group to be printed at the end of the data set. However, even though the printing of the message groups is inhibited by PIMSG=NO, the associated error still increments the message count. (A message group consists of a primary message and variable number of informational messages that result from a single error.)

In the case that multiple transmissions have been specified for a single data set (user-specified multiple copies), the message count would apply on a per copy basis. If the specified number of errors are discovered during the printing of any copy, the subject copy is terminated, and the remaining copies are not printed.

Defaults

If you do not code the PIMSG parameter, the PIMSG specification from the PSF PRINTDEV statement applies. If not specified in the PRINTDEV statement, the default is PIMSG=(YES,16).

Relationship to Other Statements

You can request a dump of specific PSF control blocks, by specifying the DUMP parameter on the PRINTDEV statement of the PSF start-up procedure (see *Print Services Facility System Programmer's Guide for MVS*). This dump can be conditional upon a specific PSF message identifier.

Turning error messages off with PIMSG=NO does not inhibit this dump facility.

For example, if PIMSG=(NO,5) is specified and the fifth qualifying error is also the error designated by the DUMP parameter on the PRINTDEV statement, PSF captures the dump information before terminating the printing of the current data set.

Examples of the PIMSG Parameter

```
//OUTDS2 OUTPUT DATACK=UNBLOCK,PIMSG=(YES,0)
```

In this example, regardless of how many message-generating errors are detected by PSF or reported to PSF by the printer, the printing of the current data set continues to completion or until a terminating error is encountered. All the messages are printed by the system.

```
//OUTDS2 OUTPUT DATACK=UNBLOCK,PIMSG=(NO,5)
```

In this example, after five message-generating errors are detected by PSF or reported to PSF by the printer, the printing of the current data set is terminated. Only the last message group is printed by the system.

PRMODE Parameter

Parameter Type: Keyword, optional

Purpose: Use the PRMODE parameter to identify the process mode required to print the sysout data set. JES schedules the data set to a printer that can operate in the specified mode.

For a list of valid process modes, contact your system programmer.

Syntax:

```
PRMODE={LINE       }
       {PAGE       }
       {process-mode}
```

Subparameter Definition

LINE
Indicates that the data set is to be scheduled to a line-mode printer.

PAGE
Indicates that the data set is to be scheduled to a page-mode printer.

process-mode
Specifies the required process mode. The process-mode is 1 through 8 alphanumeric characters.

For an NJE-transmitted data set, use PRMODE to request specific processing without having to obtain output classes for the node that processes the data set.

Defaults

If you do not code the PRMODE parameter, JES schedules output processing as follows:

- If the sysout data set does not contain page-mode formatting controls, the process mode of line is given to the data set.
- If the sysout data set contains page-mode formatting controls, the process mode of page is given to the data set.

Printing a Line-Mode Data Set Using PSF

To print a line-mode data set using the Print Services Facility (PSF) and the enhanced capabilities of the 3800 Model 3, 6 and 8, code PRMODE=PAGE. The PSF formats this line-mode data set using the installation's default values for PAGEDEF and FORMDEF defined in the PSF cataloged procedure; if these defaults are unsatisfactory, code the PAGEDEF and FORMDEF parameters on the OUTPUT JCL statement.

OUTPUT JCL: PRMODE

Example of the PRMODE Parameter

```
//DS18    OUTPUT   PRMODE=LINE
```

In this example, JES schedules the sysout data set to a printer with a process mode of line.

PRTY Parameter

Parameter Type: Keyword, optional

Purpose: Use the PRTY parameter to specify the priority at which the sysout data set enters the output queue. A data set with a higher priority is printed sooner.

Syntax:

```
PRTY=nnn
```

Subparameter Definition

nnn
Specifies the initial priority. nnn is a decimal number from 0 through 255; 0 is the lowest priority while 255 is the highest.

Defaults

If you do not code the PRTY parameter, JES3 uses an installation default specified at initialization. JES2 uses a priority that is calculated for all output.

Overrides

In JES2 systems, the installation can specify at JES2 initialization that JES2 is to ignore the OUTPUT JCL PRTY parameter.

In JES3 systems, the OUTPUT JCL PRTY parameter is ignored for JES3 networking.

Example of the PRTY Parameter

```
//PRESRPT OUTPUT PRTY=200,FORMS=TOPSEC
```

In this example, JES prints one copy of the president's report, PRESRPT, on forms named TOPSEC. Because a priority of 200 is specified, the report is probably printed immediately after entering the output queue.

THRESHLD Parameter

Parameter Type: Keyword, optional, JES3 only

Purpose: Use the THRESHLD parameter to specify the maximum size for the sysout data set. JES3 calculates the sysout data set size as the number of records multiplied by the number of copies requested. When this size exceeds the THRESHLD value, JES3 creates a new unit of work, on a data set boundary, and queues it for printing. Consequently, copies of the sysout data set may be printed simultaneously by different printers.

Use the THRESHLD parameter for jobs that generate many large data sets or many copies of one large data set.

Note: THRESHLD is supported only on JES3 systems.

Syntax:

```
THRESHLD=limit
```

Subparameter Definition

limit
Specifies the maximum number of records for a single sysout data set. limit is a decimal number from 1 through 99999999.

Defaults

If you do not code the THRESHLD parameter, JES3 uses an installation default specified at initialization.

Example of the THRESHLD Parameter

```
//STEPA   EXEC    PGM=RPTWRT
//SYSDS3  OUTPUT  DEFAULT=YES,THRESHLD=10000
//RPT1    DD      SYSOUT=A,COPIES=10
//RPT2    DD      SYSOUT=A,COPIES=2
//RPT3    DD      SYSOUT=A,COPIES=5
```

In this example, the report data sets, RPT1, RPT2, and RPT3, are processed in sysout class A. All three DD statements implicitly reference the step-level default OUTPUT JCL statement SYSDS3; therefore, the THRESHLD value specified in the OUTPUT JCL statement applies to the three reports combined. JES3 is to print the following:

Copies	Data Set	Records in Data Set	Total Records
10	RPT1	1000	10000
2	RPT2	2000	4000
5	RPT3	500	2500
		Total	16500

Because the total exceeds the THRESHLD limit, JES3 divides the sysout data sets into two units of work. RPT1 is printed as one unit, and the other two data sets are printed together as another unit. If the THRESHLD limit had been 20000, all three data sets would have been printed as one unit of work.

TRC Parameter

Parameter Type: Keyword, optional

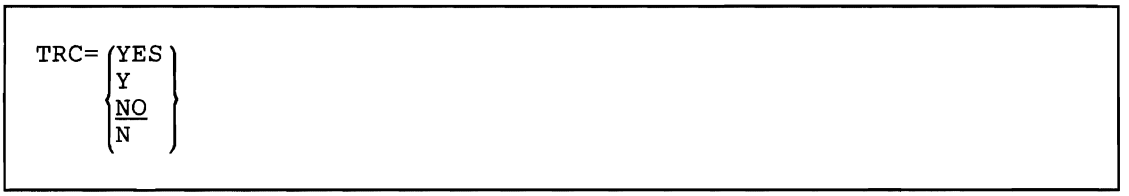
Purpose: Use the TRC parameter to specify whether the logical record for each output line in the sysout data set contains table reference character (TRC) codes or not. The TRC code identifies which table-name in the CHARS parameter is to be used to print the record.

If present, a TRC code in the output line record is:

- The first byte, if a carriage control character is not used.
- The second byte, immediately following the carriage control character, if used.

Note: TRC is supported only on JES3 systems. However, in a JES2 system, TRC can be specified for a data set processed in a functional subsystem address space (FSA); JES2 passes the TRC parameter value to the FSA. Thus, TRC can be specified for a data set printed on a 3800 Printing Subsystem Model 3, 6, and 8 by the Print Services Facility (PSF).

Syntax:



Subparameter Definition

YES
Indicates that the data set contains TRC codes. This subparameter can also be coded as Y.

NO
Indicates that the data set does not contain TRC codes. This subparameter can also be coded as N.

Defaults

If you do not code the TRC parameter, the default is NO.

Relationship to Other Parameters

A table reference character for the entire data set can be specified in the OUTPUT JCL MODIFY parameter.

Example of the TRC Parameter

```
//WRTR   JOB     ACNO77,MAEBIRD,MSGCLASS=B
//DS23   OUTPUT  DEFAULT=YES,FORMS=STD,CONTROL=PROGRAM,TRC=YES
//STEP1  EXEC    PGM=DLYRPT
//DAILY  DD      SYSOUT=A,CHARS=(GT12,GB12,GI12)
```

In this example, sysout DD statement DAILY implicitly references the default job-level OUTPUT JCL statement DS23. This OUTPUT JCL statement directs JES3 to print the daily report on standard forms. The sysout data set defined by DD statement DAILY contains carriage control characters in the first character of each logical record and a TRC code in the second character. The TRC characters in the records are 0 to use the font GT12; 1 to use GB12; and 2 to use GI12.

UCS Parameter

Parameter Type: Keyword, optional

Purpose: Use the UCS parameter to identify:

- The universal character set (UCS) image JES is to use in printing the sysout data set.
- A print train (print chain or print band) JES is to use in printing the sysout data set on an impact printer.
- A character-arrangement table for the sysout data set printed on a 3800 Printing Subsystem in a JES2 system. In this use, the UCS parameter acts like a CHARS parameter.

The UCS image specifies the special character set to be used. JES loads the image into the printer's buffer. The UCS image is stored in SYS1.IMAGELIB. IBM provides the special character set codes in Figure 17-2.

References: For more information on the UCS parameter, see *System-Data Administration.*

Syntax:

```
UCS=character-set-code
```

Subparameter Definition

character-set-code
Identifies a universal character set. The character-set-code is 1 through 4 alphanumeric or national ($, #, @) characters. See Figure 17-2 for IBM standard special character set codes.

Defaults

If you do not code the UCS parameter, the system checks the UCS image in the printer's buffer; if it is a default image, as indicated by its first byte, JES uses it. If it is not a default image, JES loads the UCS image that is the installation default specified at JES initialization.

On an impact printer, if the chain or train does not contain a valid character set, JES asks the operator to specify a character set and to mount the corresponding chain or train.

1403	3203 Model 5	3211	Characteristics
AN	AN	A11	Arrangement A, standard EBCDIC character set, 48 characters
HN	HN	H11	Arrangement H, EBCDIC character set for FORTRAN and COBOL, 48 characters
		G11	ASCII character set
PCAN	PCAN		Preferred alphanumeric character set, arrangement A
PCHN	PCHN		Preferred alphanumeric character set, arrangement H
PN	PN	P11	PL/I alphanumeric character set
QN	QN		PL/I preferred alphanumeric character set for scientific applications
QNC	QNC		PL/1 preferred alphanumeric character set for commercial applications
RN	RN		Preferred character set for commercial applications of FORTRAN and COBOL
SN	SN		Preferred character set for text printing
TN	TN	T11	Character set for text printing, 120 characters
XN			High-speed alphanumeric character set for 1403, Model 2
YN			High-speed preferred alphanumeric character set for 1403, Model N1

Note: Where three values exist (for the 1403, 3211, and 3203 Model 5 printers), code any one of them. JES selects the set corresponding to the device on which the data set is printed.

Not all of these character sets may be available at your installation. Also, an installation can design character sets to meet special needs and assign a unique code to them. Follow installation procedures for using character sets.

Figure 17-2. Special Character Sets for the 1403, 3203 Model 5, and 3211 Printers

Overrides

For printing on a printer with the UCS feature, a UCS parameter on the sysout DD statement overrides the OUTPUT JCL UCS parameter. For printing on a 3800, a CHARS parameter on the sysout DD statement or the OUTPUT JCL statement overrides all UCS parameters.

For a data set scheduled to the Print Services Facility (PSF), the PSF uses the following parameters, in override order, to select the font list:

1. Font list in the library member specified by an OUTPUT JCL PAGEDEF parameter.
2. DD CHARS parameter.
3. OUTPUT JCL CHARS parameter.
4. DD UCS parameter.
5. OUTPUT JCL UCS parameter.
6. JES installation default for the device.
7. Font list on the PAGEDEF parameter in the PSF cataloged procedure.

See "PAGEDEF Parameter" on page 17-51 for more information.

Using Special Characters Sets

To use a special character set, SYS1.IMAGELIB must contain an image of the character set, and the chain or train for the character set must be available. IBM provides standard special character sets, and the installation may provide user-designed special character sets.

OUTPUT JCL: UCS

Example of the UCS Parameter

```
//PRTDS9 OUTPUT UCS=A11
```

In this example, JES uses standard EBCDIC character set arrangement A, with 48 characters, to print the sysout data set on a 3211 printer.

WRITER Parameter

Parameter Type: Keyword, optional

Purpose: Use the WRITER parameter to name an external writer to process the sysout data set rather than JES. An external writer is an IBM- or installation-written program.

References: For information about external writers, see *SPL: System Modifications.*

Syntax:

```
WRITER=name
```

Subparameter Definition

name
Identifies the member name (1 to 8 alphanumeric characters) of an installation-written program in the system library that the external writer loads to write the output data set.

Do not code INTRDR or STDWTR (and for JES3, NJERDR) as the writer name. These names are reserved for JES.

Defaults

If you do not code the WRITER parameter, the installation's job entry subsystem processes the sysout data set.

Overrides

The writer-name subparameter of the SYSOUT parameter on the sysout DD statement overrides the OUTPUT JCL WRITER parameter.

Relationship to Other Parameters

For JES3, you can code the OUTPUT JCL DEST=nodename parameter with the WRITER=name parameter; however, do not code DEST=nodename.userid with WRITER=name.

Starting an External Writer

When a statement supplying processing options for a sysout data set specifies an external writer, the writer must be started before it can print or punch the data set. The writer is started by a system command from the operator or in the input stream. If the writer is not started before the job produces the sysout data set, the data set is retained until the writer is started.

OUTPUT JCL: WRITER

Example of the WRITER Parameter

```
//MYOUT  JOB    ACCT928,MAEBIRD,MSGCLASS=B
// START XWTR
//MYDS   OUTPUT WRITER=MYPGM
//STEP1  EXEC   PGM=REPORT
//RPT1   DD     SYSOUT=A,OUTPUT=*.MYDS
```

In a JES2 system, the second statement is a JCL command statement to start the IBM-supplied external writer. This writer is a cataloged procedure in SYS1.PROCLIB. The sysout DD statement RPT1 explicitly references OUTPUT JCL statement MYDS, which specifies that the program MYPGM is to be loaded by XWTR and process the sysout data set.

```
//**START XWTR
//MYOUT  JOB    ACCT928,MAEBIRD,MSGCLASS=B
//MYDS   OUTPUT WRITER=MYPGM
//STEP1  EXEC   PGM=REPORT
//RPT1   DD     SYSOUT=A,OUTPUT=*.MYDS
```

This example is for a JES3 system.

Chapter 18. PEND Statement

Purpose: Use the PEND statement to mark the end of an in-stream procedure.

Syntax:

```
//[name]  PEND  [comments]
```

The PEND statement consists of the characters // in columns 1 and 2 and three fields: name, operation (PEND), and comments.

Do not continue a PEND statement.

Name Field

A name is optional on the PEND statement. If used, code it as follows:

- Each name must be unique within the job.
- The name must begin in column 3.
- The name is 1 through 8 alphanumeric or national ($, #, @) characters.
- The first character must be alphabetic or national ($, #, @).
- The name must be followed by at least one blank.

If a name is not coded, column 3 must be blank.

Operation Field

The operation field consists of the characters PEND and must be preceded and followed by at least one blank. It can begin in any column.

Comments Field

The comments field follows PEND after at least one intervening blank.

PEND

Location in the JCL

A PEND statement follows the statements of an in-stream procedure. Never place a PEND statement in a cataloged procedure.

Examples of the PEND Statement

```
//PROCEND1 PEND THIS STATEMENT IS REQUIRED FOR IN-STREAM PROCEDURES
```

This PEND statement contains a comment.

```
// PEND
```

This PEND statement contains only // and the operation field with the necessary blanks.

Chapter 19. PROC Statement

Purpose: The PROC statement **must** mark the beginning of an in-stream procedure and, optionally, may mark the beginning of a cataloged procedure. For either procedure, the PROC statement can assign default values to symbolic parameters, if coded, in the procedure.

Syntax:

```
For a cataloged procedure:
         //[name]  PROC [parameter  [comments]]
         //[name]  PROC

For an in-stream procedure:
         //name  PROC [parameter  [comments]]
         //name  PROC
```

A PROC statement consists of the characters // in columns 1 and 2 and four fields: name, operation (PROC), parameter, and comments.

Multiple Parameters: When more than one parameter is coded, separate parameters by commas. For example, //P1 PROC PARM1=OLD,PARM2=222001.

Special Characters: When a parameter value contains special characters, enclose the value in apostrophes. The enclosing apostrophes are not considered part of the value. For example, //P2 PROC PARM3='3400-6'.

Code each apostrophe that is part of a value as two consecutive apostrophes. For example, //P3 PROC PARM4='O''DAY'.

Continuation onto Another Statement: End each statement with a comma after a complete parameter. For example:

```
//P4 PROC  PARM5=OLD,PARM6='SYS1.LINKLIB(P40)',
//       PARM7=SYSDA,PARM8='(CYL,(10,1))'
```

PROC

Name Field

A name is required on a PROC statement in an in-stream procedure and is optional on a PROC statement in a cataloged procedure. Code it as follows:

- Each name must be unique within the job.
- The name must begin in column 3.
- The name is 1 through 8 alphanumeric or national ($, #, @) characters.
- The first character must be alphabetic or national ($, #, @).
- The name must be followed by at least one blank.

If a name is not coded, column 3 must be blank.

Operation Field

The operation field consists of the characters PROC and must be preceded and followed by at least one blank. It can begin in any column.

Parameter Field

The parameters on a PROC statement assign default values to symbolic parameters on procedure statements. An in-stream PROC statement requires parameters only if the procedure contains symbolic parameters. See "Symbolic Parameters" on page 5-8 for details on symbolic parameters and on how to assign values to them.

If coded, the parameter field must be preceded and followed by at least one blank.

Comments Field

The comments field follows the parameter field after at least one intervening blank. Do not code comments unless you code the parameter field.

Overrides

To override a default parameter value on a PROC statement, code the same parameter on the EXEC statement that calls the procedure.

Location in the JCL

A PROC statement must begin all in-stream procedures and may begin a cataloged procedure. An in-stream procedure must appear in the same job before the EXEC statement that calls it. A cataloged procedure appears in a procedure library, usually SYS1.PROCLIB.

Examples of the PROC Statement

```
//DEF      PROC  STATUS=OLD,LIBRARY=SYSLIB,NUMBER=777777
//NOTIFY   EXEC  PGM=ACCUM
//DD1      DD    DSNAME=MGMT,DISP=(&STATUS,KEEP),UNIT=3400-6,
//             VOLUME=SER=888888
//DD2      DD    DSNAME=&LIBRARY,DISP=(OLD,KEEP),UNIT=3350,
//             VOLUME=SER=&NUMBER
```

Three symbolic parameters are defined in this cataloged procedure: &STATUS, &LIBRARY, and &NUMBER. Values are assigned to the symbolic parameters on the PROC statement. These values are used when the procedure is called and values are not assigned to the symbolic parameters on the calling EXEC statement.

```
//CARDS  PROC
```

This PROC statement can be used to mark the beginning of an in-stream procedure named CARDS.

Chapter 20. XMIT JCL Statement

Support for the XMIT JCL Statement:

- The XMIT JCL statement is supported only on JES3 systems.
- For JES2, the XMIT JCL statement is not supported. Use the JES2 /*XMIT statement to transmit records.
- For TSO, the XMIT JCL statement is supported only with TSO Extensions Release 3 or later. With earlier TSO releases, network jobs submitted through TSO that use the XMIT JCL statement are run locally.

Purpose: Use the XMIT JCL statement to transmit records from an MVS JES3 node to a JES3 node, a JES2 node, a VSE/POWER node, or a VM/RSCS node.

The sending system does not process or check the records for validity except when the JCL is processed by an internal reader (such as with TSO submit processing). In this case, the system recognizes /*EOF and /*DEL as internal reader control statements and errors can occur on the sending system if /*EOF or /*DEL are included in the XMIT JCL stream.

To transmit /*EOF and /*DEL statements as part of the XMIT JCL stream, replace /* with two substitute characters and identify the substitute characters on the SUBCHARS parameter. Prior to transmission, the sending system converts the two substitute characters to /*. The receiving (execution) system can then process the /*EOF and /*DEL statements as internal reader control statements.

Do not nest XMIT JCL statements. That is, do not include an XMIT JCL statement between an XMIT JCL statement and its delimiter.

The system builds network job header and trailer records from information on the JOB statement and any //*NETACCT statements, if included, preceding the XMIT JCL statement. Then the system transmits all the records between the XMIT JCL statement and a delimiter statement.

The records can consist of a job input stream, an in-stream DD * or DD DATA data set, or any job definition statements recognized by the destination node. If the records are a job input stream, and the destination node can process the JCL, the transmitted input stream is executed at the destination node.

The records end when the system finds one of the following delimiters:

- /* in the input stream, if a DLM parameter is not coded on this XMIT JCL statement. (Refer to Chapter 12. Delimiter Statement for an explanation of /*.)

  From TSO only, TSO inserts /* at the end-of-file if the default delimiter is not supplied.

XMIT JCL

- The two-character delimiter specified by a DLM parameter on this XMIT JCL statement.

Syntax:

```
//[name]  XMIT  parameter[,parameter]...  [comments]
```

The XMIT JCL statement consists of the characters // in columns 1 and 2 and four fields: name, operation (XMIT), parameter, and comments.

Name Field

A name is optional on the XMIT JCL statement. If used, code it as follows:

- Each name must be unique within the job.
- The name must begin in column 3.
- The name is 1 through 8 alphanumeric or national ($, #, @) characters.
- The first character must be alphabetic or national ($, #, @).
- The name must be followed by at least one blank.

Operation Field

The operation field consists of the characters XMIT and must be preceded and followed by at least one blank. It can begin in any column.

Parameter Field

The XMIT JCL statement contains only keyword parameters. A DEST parameter is required; the DLM and SUBCHARS parameters are optional. If your JCL is to be processed by an internal reader and /*EOF or /*DEL is part of the XMIT JCL stream, you must code the SUBCHARS parameter.

You can code the keyword parameters in any order in the parameter field.

KEYWORD PARAMETERS	VALUES	PURPOSE
DEST = nodename[.vmuserid] See page 20-5	nodename: 1 - 8 alphanumeric or $, #, @ characters vmuserid: 1 - 8 alphanumeric or $, #, @ characters	Identifies the destination for all following records until a delimiter stops transmission of the records.
DLM = delimiter See page 20-6	delimiter: 2 alphanumeric, $, #, @, or special characters	Specifies a delimiter to stop the transmission of records.
SUBCHARS = substitute See page 20-8	substitute: 2 alphanumeric, $, #, @, or special characters	Specifies a substitute for internal reader control statements.

Comments Field

The comments field follows the parameter field after at least one intervening blank.

Location in the JCL

Place the XMIT JCL statement after a JOB statement and any //*NETACCT statements. (Other JES3 JCL statements between the JOB and XMIT JCL statements are not supported and can cause unpredictable results.) The JOB statement must be valid for the submitting location.

Do not place any other MVS JCL statements between the JOB statement and the XMIT JCL statement. If any of these statements intervene, the system terminates the job.

Error on XMIT JCL Statement

If the system finds an error on the XMIT JCL statement before a specified DLM parameter, all jobs in the batch are flushed.

If the system finds an error on the XMIT JCL statement after a specified DLM parameter, the network job is flushed and local processing starts at the statement following the specified delimiter.

Examples of the XMIT JCL Statement

```
//JOBA    JOB   25FA64,'KEN KAHN'
//X1      XMIT  DEST=KGNMVS45
          .
          .
         (records to be transmitted)
          .
/*
//JOBB    JOB   ...
          .
```

In this example, the records between the XMIT JCL statement and the delimiter statement (/* in columns 1 and 2) are transmitted to the node named KGNMVS45.

```
//JOBC    JOB  PW19,'DEPT 53'
//X2      XMIT DEST=POKVMDD3.MVSGST34,DLM=AA
          .
          .
/*
         (records to be transmitted)
/*EOF
/*DEL
          .
          .
AA
//JOBB    JOB   ...
          .
```

In this example, processing is **not** through an internal reader on the sending system. The records between the XMIT JCL statement and the delimiter statement, which must contain AA in columns 1 and 2 as specified in the DLM parameter, are transmitted to the system, MVSGST34, running on the VM system at the node named POKVMDD3.

```
//JOBE    JOB   NS37,'NYC BX'
//X3      XMIT  DEST=SANFRAN,DLM=AA,SUBCHARS='/+'
          .
          .
         (records to be transmitted)
          .
          .
/+EOF
          .
          .
/+DEL
AA
//JOBF    JOB   ...
          .
```

In this example, the JCL is processed through an internal reader on the sending system. The records between the XMIT JCL statement and the delimiter statement, which must contain AA in columns 1 and 2 as specified in the DLM parameter, are transmitted to the node named SANFRAN.

To transmit the /*EOF and /*DEL internal reader control statements, /* is replaced by /+ in columns 1 and 2 on both statements in the XMIT JCL stream and SUBCHARS='/+' is coded on the XMIT statement. The sending system does not recognize /+EOF and /+DEL as internal reader statements. Then prior to transmission, the sending system converts /+ to /* and sends /*EOF and /*DEL to the receiving node, which can then process the internal reader control statements.

DEST Parameter

Parameter Type: Keyword, required

Purpose: Use the DEST parameter to specify a destination for the following input stream records. The DEST parameter can send the records to a node or, for a node that is a VM system, to a guest system running on the virtual machine.

Syntax:

```
DEST=nodename

DEST=nodename.vmuserid
```

Subparameter Definition

nodename
Identifies the destination node. The nodename identifies a JES2 system, a JES3 system, a VSE/POWER node, or a VM system. The nodename is 1 through 8 alphanumeric or national ($, #, @) characters specified during JES initialization. If the requested node is the same as the submitting node, the records following the XMIT JCL statement are processed by the local system.

vmuserid
Identifies a destination guest system running in a virtual machine. The vmuserid is 1 through 8 alphanumeric or national ($, #, @) characters.

Examples of the DEST Parameter

```
//TRANS XMIT DEST=LAXSYS
```

This example sends the following records to a node named LAXSYS.

```
//SEND  XMIT DEST=VMSYS3.GUEST7
```

This example sends the following records to a guest system, named GUEST7, running in the VM system at the node named VMSYS3.

DLM Parameter

Parameter Type: Keyword, optional

Purpose: Use the DLM parameter to specify a delimiter to stop transmission of input stream records. When the DLM parameter assigns a delimiter other than the standard delimiter (/* in columns 1 and 2), the records can include the standard delimiter.

If you use the DLM delimiter to define a delimiter, be sure to terminate the records with the specified DLM characters. Otherwise, all jobs between the XMIT JCL statement and the end-of-file are flushed.

From TSO only, TSO inserts /* at the end-of-file if the default delimiter is not supplied.

Syntax:

```
DLM=delimiter
```

- If the specified delimiter contains any special characters, enclose the delimiter in apostrophes. In this case, a special character is any character that is neither alphanumeric nor national ($, #, @).
- If the delimiter contains an ampersand or an apostrophe, code each ampersand or apostrophe as two consecutive ampersands or apostrophes and enclose the delimiter in apostrophes. Each pair of consecutive ampersands or apostrophes counts as one character.

Subparameter Definition

delimiter
Specifies two characters that indicate the end of this data set in the input stream.

Default

If you do not specify a DLM parameter, the default is the standard /* delimiter statement.

Invalid Delimiters

If the delimiter is not two characters, the system terminates the job and does not transmit any records.

Examples of the DLM Parameter

```
//XX   XMIT   DEST=NYCNODE,DLM=AA
       .
       .
       (records to be transmitted)
       .
AA
```

The DLM parameter assigns the characters AA as the delimiter for the in-stream records to be transmitted.

```
//XY   XMIT   DEST=ATL,DLM='A+'
//XZ   XMIT   DEST=BOST,DLM='&&7'
//XW   XMIT   DEST=CHI,DLM='B'''
```

These examples specify delimiters of A+, &7, and B'.

SUBCHARS Parameter

Parameter Type: Keyword, optional

Purpose: Use the SUBCHARS parameter to specify a substitute (consisting of two characters) for the first two characters of /*EOF and /*DEL internal reader control statements. The substitute characters on the internal reader control statements must be in columns 1 and 2.

You can use the SUBCHARS parameter for any XMIT JCL job. However, SUBCHARS is required if you want to transmit internal reader control statements (/*EOF and /*DEL) and the job is processed by an internal reader on the sending system. Note that the system recognizes /*EOF and /*DEL as internal reader control statements and errors can occur on the sending system if /*EOF or /*DEL are included in the XMIT JCL stream.

To transmit internal reader control statements, replace /* on the /*EOF and /*DEL statements in the records to be transmitted with two substitute characters and identify the substitute characters on the SUBCHARS parameter. Prior to transmission, the system converts the substitute characters to /* and sends /*EOF and /*DEL to the receiving node for processing.

Reference: The internal reader is described in *SPL: JES3 Initialization and Tuning* and *SPL: System Modifications.*

Syntax:

```
SUBCHARS=substitute
```

- If the specified substitute contains any special characters, enclose the substitute in apostrophes. In this case, a special character is any character that is neither alphanumeric nor national ($, #, @).
- If the substitute contains an ampersand or an apostrophe, code each ampersand or apostrophe as two consecutive ampersands or apostrophes and enclose the substitute in apostrophes. Each pair of consecutive ampersands or apostrophes counts as one character.

Subparameter Definition

substitute
Specifies two characters that indicate the substitute characters for the first two characters of internal reader control statements. The substitute characters apply only to internal reader statements.

Default

There is no default for SUBCHARS.

Invalid Substitute

If the substitute is not two characters, the system terminates the job and does not transmit any records.

Examples of the SUBCHARS Parameter

```
//XX  XMIT  DEST=NYCNODE,SUBCHARS=MV
       .
       .
       (records to be transmitted)
MVEOF
       .
```

The SUBCHARS parameter identifies the characters MV as the substitute for the first two characters of the internal reader control statement to be transmitted. Prior to transmission, the system converts the MV substitute characters to /* and sends /*EOF to the receiving node for processing.

```
//XY  XMIT  DEST=ATL,SUBCHARS='A+'
//XZ  XMIT  DEST=BOST,SUBCHARS='&&7'
//XW  XMIT  DEST=CHI,SUBCHARS='B'''
```

These examples specify substitutes of A+, &7, and B'.

Chapter 21. JES2 Control Statements

Code JES2 control statements with JCL statements to control the input and output processing of jobs. The rules for coding in Chapter 3, "Format of Statements" and Chapter 4, "Syntax of Parameters" apply to the JES2 control statements.

Location in the JCL

Place JES2 control statements, except the command and /*PRIORITY statements, after the JOB statement and its continuations. JES2 ignores JES2 control statements, except the command and /*PRIORITY statements, that appear before the JOB statement or between continued JOB statements.

Do not include JES2 control statements in a cataloged or in-stream procedure. JES2 ignores JES2 control statements in a procedure.

Internal Reader

Use the following control statements when submitting jobs to the internal reader. The internal reader is described in *SPL: JES2 Initialization and Tuning* and *SPL: System Modifications.*

/*DEL
/*EOF
/*PURGE
/*SCAN

JES2 Command Statement

Purpose: Use the command statement to enter a JES2 operator command through the input stream, the internal reader, or the system console.

JES2 usually executes an in-stream command as soon as it is read. Therefore, the command will **not** be synchronized with the execution of any job or step in the input stream. To synchronize a command with the job processing, tell the operator the commands you want and when they should be issued, and let the operator enter them from the console.

Examples in this book illustrate the format for commands entered through the input stream. Commands entered through an operator console should not have /* in columns 1 and 2.

References: For more information on the command statement and the JES2 verbs and operands, see *JES2 Commands*.

Syntax:

```
/*$command-verb,operand[,operand]...      [N]
```

The JES2 command statement consists of:

- The characters /* in columns 1 and 2.
- $ or a character chosen by the installation in column 3.
- The command verb beginning in column 4.
- A comma.
- Operands up through column 71.
- N in column 72 if JES2 is **not** to write the command on the operator console.
- Blanks in columns 73 through 80. JES2 ignores these columns.

Do not continue command statements from one statement to the next, instead code as many command statements as you need.

Parameter Definition

command-verb
Specifies the operator command that JES2 is to perform. You can enter the following JES2 commands in the input stream.

$A	$E	$I	$O	$T
$B	$F	$L	$P	$TRACE
$C	$G	$M	$R	$V
$D	$H	$N	$S	$Z

operand
Specifies options for the command.

N in column 72
Indicates that JES2 is not to repeat the command on the operator console.

Location in the JCL

Place JES2 command statements before jobs being entered through the input stream. JES2 ignores any JES2 command statements within a job.

If a job contains a JES2 /*XMIT statement, place the command statement:

- Before the /*XMIT statement if you want JES2 to process and display the command at the input node only.
- After the /*XMIT statement if you want JES2 to process and display the command at the execution node only.

Examples of the Command Statement

```
/*$SI3-5
```

This command statement starts initiators three through five. The command is $S and the operand is I3-5. JES2 executes the command immediately and repeats the command on the operator console.

```
/*$TRDR1,H=Y
```

In response to this command, JES2 places all jobs being read by reader 1 in a hold status. If a job contains a JES2 /*ROUTE XEQ or /*XEQ statement that specifies an execution node different from the input node, JES2 holds the job at the execution node, not the input node.

/*JOBPARM Statement

Purpose: Use the /*JOBPARM statement to specify job-related parameters for JES2.

Syntax:

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

The parameters are:

      {BURST}={Y}
      {B    } {N}

      {BYTES}=nnnnnn
      {M    }

      {CARDS}=nnnnnnn
      {C    }

      {COPIES}=nnn
      {N     }

      {FORMS}={xxxxxxxx}
      {F    } {STD     }

      {LINECT}=nnn
      {K     }

      {LINES}=nnnn
      {L    }

      {NOLOG}
      {J    }

      {PAGES}=nnnnn
      {G    }

      {PROCLIB}=ddname
      {P      }

      {RESTART}={Y}
      {E      } {N}

      {ROOM}=xxxx
      {R   }

      {SYSAFF}={*                    }
      {S     } {(*[,IND])            }
               {ANY                  }
               {(ANY[,IND])          }
               {cccc                 }
               {(cccc[,IND])         }
               {(cccc[,cccc]...)     }
               {((cccc[,cccc]...)[,IND])}

      {TIME}=nnnn
      {T   }
```

The /*JOBPARM statement consists of the characters /* in columns 1 and 2, JOBPARM in columns 3 through 9, a blank in column 10, and parameters in columns 11 through 71. JES2 ignores columns 72 through 80.

Do not continue a /*JOBPARM statement. Instead, code as many /*JOBPARM statements as necessary in an input stream.

Code any number of the above parameters on a single /*JOBPARM statement.

Parameter Definition

BURST=Y
BURST=N

Specifies the default burst characteristic of all sysout data sets that JES2 produces for this job. BURST applies only when the data set is directed to a 3800 Printing Subsystem equipped with a burster-trimmer-stacker.

Y

Requests that the 3800 output is to be burst into separate sheets.

N

Requests that the 3800 output is to be in a continuous fanfold.

BYTES=nnnnnn

Specifies the maximum output, in thousands of bytes, the system is to produce from this job. The nnnnnn is 1 through 6 decimal numbers from 0 through 999999. When nnnnnn bytes are reached, JES2 gives control to a user exit routine and the job might or might not be terminated.

CARDS=nnnnnnn

Specifies the maximum number of output cards to be punched for this job's sysout data sets. The nnnnnnn is 1 through 7 decimal numbers from 0 through 9999999. When nnnnnnn cards are reached, JES2 gives control to a user exit routine and the job might or might not be terminated.

COPIES=nnn

Specifies how many copies of the spool lines or bytes for this job's sysout data sets are to be printed or punched. The nnn is 1 through 3 decimal numbers from 1 through 255. An installation can reduce the upper limit of this value during JES2 initialization.

The COPIES parameter is ignored and only one copy is produced if any of the following is true:

- FREE=CLOSE is coded on the DD statement for the output data set.
- HOLD=YES is coded on any sysout DD statement in the job.
- The output class of the sysout data set is a held class, and the message class is also a held class. The message class is specified in the JOB statement MSGCLASS parameter.

FORMS = xxxxxxxx
FORMS = STD
Specifies the print and/or punch forms JES2 is to use for sysout data sets for which FORMS is not specified on the DD statement or on a JES2 /*OUTPUT statement.

xxxxxxxx
Identifies the print or punch forms. The xxxxxxxx is 1 through 8 alphanumeric or national ($, #, @) characters.

STD
Indicates that JES2 is to use the default specified at JES2 initialization.

LINECT = nnn
Specifies the maximum number of lines that JES2 is to print on each output page for this job's sysout data sets. The nnn is 1 through 3 numbers from 0 through 255.

If you code LINECT = 0, JES2 does not eject to a new page when the number of output lines exceeds the page limit that the installation specified during JES2 initialization.

The LINECT parameter on the /*OUTPUT statement overrides LINECT on the /*JOBPARM statement and the linect value in the accounting information parameter of the JOB statement.

LINES = nnnn
Specifies the maximum output, in thousands of lines, that JES2 is to place in the spool data sets for this job's sysout data sets. The nnnn is 1 through 4 decimal numbers from 0 through 9999. When nnnn lines are reached, JES2 gives control to a user exit routine and the job might or might not be terminated.

The LINES parameter applies only to line-mode data. (See also the PAGES parameter.) If the sysout data set contains both line-mode and page-mode data, the lines and pages are counted separately and each checked for limit excession.

NOLOG
Requests that JES2 not print the job's hard-copy log. The job's hard-copy log contains the JES2 and operator messages about the job's processing: allocation of devices and volumes, execution and termination of job steps and the job, and disposition of data sets.

PAGES = nnnnn
Specifies the maximum number of output pages to be printed for this job's sysout data sets. The nnnnn is 1 through 5 decimal numbers from 0 through 99999. When nnnnn pages are reached, JES2 gives control to a user exit routine and the job might or might not be terminated.

The PAGES parameter applies only to page-mode data. (See also the LINES parameter.) If the sysout data set contains both page-mode and line-mode data, the pages and lines are counted separately and each checked for limit excession.

PROCLIB = ddname
Requests a JES2 procedure library by its ddname in the system procedure library, SYS1.PROCLIB. These JES2 procedure library ddnames are in the format PROCnn, where nn is 1 or 2 decimal numbers from 1 through 99. The system retrieves called cataloged procedures from the requested JES2 procedure library. If the PROCLIB parameter is omitted, or the ddname cannot be found in SYS1.PROCLIB, the procedure

library specified by the JOBCLASS initialization parameter is used. If none is found, the default, PROC00 is used.

RESTART=Y
RESTART=N
Requests one of the following, if this job is executing before a re-IPL and JES2 warm start, and the job cannot restart from a step or checkpoint.

Y
Requests that JES2 queue the job for re-execution from the beginning of the job.

N
Requests that JES2 take no special action.

Note: If you do not specify RESTART, JES2 assumes N. However, the installation may override this default in JES2 initialization parameters.

ROOM=xxxx
Indicates the programmer's room number. The xxxx is 1 through 4 alphanumeric characters. JES2 places the room number on the job's separators so that the installation can deliver the job's sysout data sets to the programmer.

SYSAFF=*
SYSAFF=(*[,IND])
SYSAFF=ANY
SYSAFF=(ANY[,IND])
SYSAFF=cccc
SYSAFF=(cccc[,IND])
SYSAFF=(cccc[,cccc]...)
SYSAFF=((cccc[,cccc]...)[,IND])
Indicates the systems that are eligible to process the job. The parameter indicates from 1 through 7 system affinities.

Note: For TSO-submitted jobs that specify NOTIFY in the JOB statement: after a job has completed execution, JES2 may change the SYSAFF specification for the job if the job executed on a processor other than the processor that the user is logged on. This is done by JES2 during output processing to allow NOTIFY processing to take place on the user's processor.

*****
Indicates the system that read the job.

ANY
Indicates any system in the JES2 multi-access spool configuration.

cccc
Indicates a specific system. cccc must be the 4-alphanumeric-character system id of one of the systems in the JES2 multi-access spool configuration. To specify more than one system, separate the system ids with commas and enclose the system id list in parentheses; for example, SYSAFF=(cccc,cccc,cccc). Repeat cccc up to the number of processors in the configuration.

IND

After any of the other SYSAFF specifications, indicates that JES2 is to use system scheduling in independent mode. When IND is coded, the subparameters must be enclosed in parentheses.

TIME = nnnn

Estimates the job execution time, in minutes of real time. The nnnn is 1 through 4 decimal numbers from 0 through 9999. If you omit a TIME parameter and a time subparameter in the JOB statement accounting information parameter, JES2 uses an installation default specified at initialization. If job execution exceeds the time, JES2 sends a message to the operator.

Overrides

- The /*JOBPARM statement parameters override the installation defaults specified at JES2 initialization.
- An OUTPUT JCL statement can override parameters on a /*JOBPARM statement.
- A JES2 /*OUTPUT statement can override parameters on a /*JOBPARM statement.
- Any /*JOBPARM statement parameter value overrides the equivalent parameter value from the JES2 accounting information on the JOB statement or from any preceding /*JOBPARM statement in this job.

Location in the JCL

Place the /*JOBPARM statement after the JOB statement.

Execution Node

JES2 normally processes /*JOBPARM statements at the node of execution.

When you place a /*JOBPARM statement **before** a /*ROUTE XEQ or /*XEQ statement, JES2 at the input node checks the /*JOBPARM statement for syntax and parameter validity. After processing the /*ROUTE XEQ or /*XEQ statement, JES2 then passes the /*JOBPARM statement to the execution node, where syntax and parameter validity are again checked.

When you place a /*JOBPARM statement **after** a /*ROUTE XEQ or /*XEQ statement, JES2 passes the /*JOBPARM to the execution node and performs all syntax and parameter validity processing at the execution node only.

COPIES Parameter in Remote Processing: In remote processing, the COPIES parameter on the /*JOBPARM statement determines the number of output copies only when the execution node is a JES2 node. The /*JOBPARM COPIES parameter is not supported by RSCS, DOS/VSE POWER, or JES3.

Example of the /*JOBPARM Statement

```
/*JOBPARM  LINES=60,ROOM=4222,TIME=50,PROCLIB=PROC03,COPIES=5
/*JOBPARM  L=60,R=4222,T=50,P=PROC03,N=5
```

The two statements specify the same parameters and values. The parameter specifications mean the following:

LINES=60 or L=60
The job's estimated output will be 60,000 lines.

ROOM=4222 or R=4222
The programmer's room is 4222. JES2 places this information in the separators for both printed and punched data sets.

TIME=50 or T=50
The job's estimated execution time is 50 minutes.

PROCLIB=PROC03 or P=PROC03
The procedure library that JES2 is to use to convert the JCL for this job is PROC03.

COPIES=5 or N=5
The estimated 60,000 lines of output will be printed five times.

/*MESSAGE Statement

Purpose: Use the /*MESSAGE statement to send messages to the operator console when JES2 reads in the job.

Syntax:

```
/*MESSAGE message
```

The /*MESSAGE statement consists of the characters /* in columns 1 and 2, MESSAGE in columns 3 through 9, a blank in column 10, and the message starting in any column from 11 through 71. JES2 ignores columns 72 through 80.

Relationship to the /*ROUTE XEQ Statement

If the /*MESSAGE statement is in a job that also contains a JES2 /*ROUTE XEQ statement:

- Placing the /*MESSAGE statement before the /*ROUTE XEQ statement directs JES2 to send the message to the operators at the input node and the execution node.
- Placing the /*MESSAGE statement after the /*ROUTE XEQ statement directs JES2 to send the message only to the operator at the execution node.

Location in the JCL

If the /*MESSAGE statement is after the JOB statement, JES2 appends the job number to the beginning of the message.

If the /*MESSAGE statement is not within a job, JES2 appends the input device name to the beginning of the message.

Example of the /*MESSAGE Statement

```
/*MESSAGE CALL DEPT 58 WHEN PAYROLL JOB IS FINISHED--EX.1946
```

JES2 sends this message to the operator console when the job is read in.

/*NETACCT Statement

Purpose: Use the /*NETACCT statement to specify an account number that is available to all the nodes in a network. JES2 uses the account number as is or translates it to local account numbers.

Syntax:

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

The /*NETACCT statement consists of the characters /* in columns 1 and 2, NETACCT in columns 3 through 9, a blank in column 10, and the network account number starting in any column from 11 through 71. JES2 ignores columns 72 through 80.

Parameter Definition

network-account-number
Specifies the job's accounting number. The network-account-number is 1 through 8 alphanumeric characters.

Defaults

If no /*NETACCT statement is specified, JES2 uses the local account number to search a table for the network account number.

Overrides

If you supply both a /*NETACCT and a local account number, JES2 uses the local account number on the input node.

Location in the JCL

Place the /*NETACCT statement after the JOB statement.

If a job contains more than one /*NETACCT statement, JES2 uses the network account number from the last statement.

JES2 ignores the /*NETACCT statement on any node other than the input node.

Example of the /*NETACCT Statement

```
/*NETACCT NETNUM10
```

JES2 transmits the network account number, NETNUM10, with the job to the destination node.

/*NOTIFY Statement

Purpose: Use the /*NOTIFY statement to direct a job's notification messages to a user.

Note: The /*NOTIFY statement does not affect where the job is executed or where output is printed or punched.

Syntax:

```
                 ┌ nodename.userid ┐
                 │ nodename:userid │
/*NOTIFY         { nodename/userid }
                 │ nodename(userid)│
                 └ userid          ┘
```

The /*NOTIFY statement consists of the characters /* in columns 1 and 2, NOTIFY in columns 3 through 8, a blank in column 10, and a parameter starting in any column from 11 through 71. JES2 ignores columns 72 through 80.

Do not code a comma, a right parenthesis, or a blank character in the nodename or userid.

Parameter Definition

nodename.userid
nodename:userid
nodename/userid
nodename(userid)

Identifies a node and a TSO or VM userid at that node. The nodename is a symbolic name defined by the installation during initialization; nodename is 1 through 8 alphanumeric or national ($, #, @) characters. The userid must be defined at the node; userid for TSO is 1 through 7 alphanumeric or national ($, #, @) characters and for VM is 1 through 8 alphanumeric or national ($, #, @) characters.

When you specify a nodename and userid, JES2 sets the origin node field in the job's network job header to the specified node, even though the job origin node may be different.

userid

Identifies a TSO or VM user. The userid for TSO is 1 through 7 alphanumeric or national ($, #, @) characters and for VM is 1 through 8 alphanumeric or national ($, #, @) characters. When you specify only a userid, JES2 assumes that the userid is at the origin node.

Overrides

The JES2 /*NOTIFY statement overrides the NOTIFY parameter on the JOB statement.

Relationship to Other Control Statements

If you submit a TSO job with a JOB statement NOTIFY parameter or the job includes a JES2 /*NOTIFY statement, do not specify IND in a JES2 /*JOBPARM SYSAFF parameter; the IND subparameter would change the mode of the job. This change would be invalid because the mode of the job must match the mode of the system at which the job is submitted.

Location in the JCL

The /*NOTIFY statement directs the notification messages of the job in which it appears; place the /*NOTIFY statement after the JOB statement and before the first EXEC statement.

Examples of the NOTIFY Statement

```
/*NOTIFY VMNODE.VMUSER
```

JES2 sends notification messages to user VMUSER on node VMNODE.

```
/*NOTIFY TSOUSER
```

JES2 sends notification messages to user TSOUSER on the job's origin node.

/*OUTPUT Statement

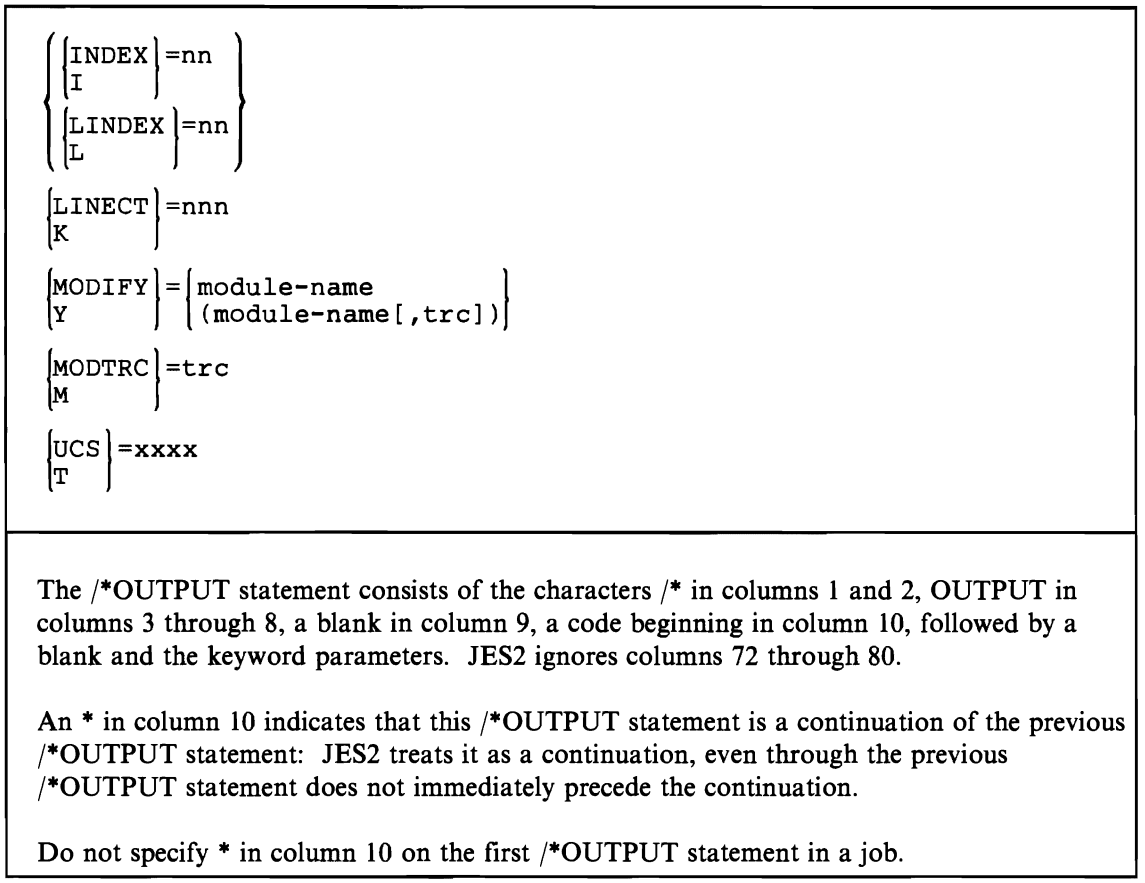
Purpose: Use the /*OUTPUT statement to specify characteristics and options for one or more sysout data sets. This statement supplies processing options in addition to and in place of the options specified on the sysout DD statement.

Note: **You should use the OUTPUT JCL statement instead of the JES2 /*OUTPUT statement because of the OUTPUT JCL statement's enhanced output processing capabilities.**

Syntax:

```
/*OUTPUT code parameter[,parameter]...

The parameters are:

{BURST }={Y}
{B     } {N}

{CHARS }={xxxx                 }
{X     } {(xxxx[,xxxx]...)     }

{CKPTLNS}=nnnnn
{E      }

{CKPTPGS}=nnnnn
{P      }

{COMPACT}=nn
{Z      }

{COPIES}={nnn                                      }
{N     } {(nnn[,(group-value[,group-value]...)])   }

{COPYG}={group-value                 }
{G    } {(group-value[,group-value]...)}

{DEST}={destination                 }
{D   } {(destination[,destination]...)}

{FCB}=xxxx
{C  }

{FLASH}={overlay-name               }
{O    } {(overlay-name[,count])     }
        {NONE                       }

{FLASHC}=count
{Q     }

{FORMS}={xxxx}
{F    } {STD }
```

```
destination is:

{LOCAL          }
{name           }
{Nnnnn          }
{NnnRmmmm       }
{NnnnRmmm       }
{NnnnnRmm       }
{nodename.userid}
{nodename:userid}
{nodename/userid}
{nodename(userid)}
{Rnnnn          }
{RMnnnn         }
{RMTnnnn        }
{Unnnn          }
```



The /*OUTPUT statement consists of the characters /* in columns 1 and 2, OUTPUT in columns 3 through 8, a blank in column 9, a code beginning in column 10, followed by a blank and the keyword parameters. JES2 ignores columns 72 through 80.

An * in column 10 indicates that this /*OUTPUT statement is a continuation of the previous /*OUTPUT statement: JES2 treats it as a continuation, even through the previous /*OUTPUT statement does not immediately precede the continuation.

Do not specify * in column 10 on the first /*OUTPUT statement in a job.

Parameter Definition

code

Identifies the /*OUTPUT statement. The code is 1 through 4 alphanumeric characters. To refer to a /*OUTPUT statement, the DD statement SYSOUT parameter must specify this code in its code-name subparameter. The referenced /*OUTPUT statement specifies processing options for the sysout data set defined in the referencing DD statement.

A code of * indicates that this /*OUTPUT statement is a continuation of the previous /*OUTPUT statement.

Note: If you specify the code-name subparameter on a DD statement SYSOUT parameter in a job or job step that contains a default OUTPUT JCL statement, JES2 uses the default OUTPUT JCL statement instead of the reference to the /*OUTPUT statement.

If more than one /*OUTPUT statement has the same code starting in column 10, JES2 uses the parameters from only the first /*OUTPUT statement.

BURST = Y
BURST = N
Indicates the default burst characteristic of all sysout data sets that JES2 produces for this job. BURST applies only when the data set is directed to a 3800 Printing Subsystem equipped with a burster-trimmer-stacker.

Y
Requests that the 3800 output is to be burst into separate sheets.

N
Requests that the 3800 output is to be in a continuous fanfold.

CHARS = xxxx
CHARS = (xxxx[,xxxx]...)
Names a character-arrangement table for all output that JES2 prints on a 3800 Printing Subsystem in this job. The xxxx is 1 through 4 alphanumeric or national ($, #, @) characters. Code one to four names.

CKPTLNS = nnnnn
Specifies the maximum number of lines or cards contained in a logical page. The nnnnn is 1 through 5 decimal numbers from 0 through 32767 for printers and 1 through 32767 for punches. The default is specified in the JES2 initialization parameter for the device.

CKPTPGS = nnnnn
Specifies the number of logical pages to be printed before the next checkpoint is taken. The nnnnn is 1 through 5 decimal numbers from 1 through 32767. The default is specified in the JES2 initialization parameter for the device.

COMPACT = nn
Specifies a compaction table for JES2 to use when sending this sysout data set, which must be a systems network architecture (SNA) data set, to a SNA remote terminal.

Note: The COMPACT parameter has no effect on compaction for NJE sessions; it applies only to SNA RJE sessions.

COPIES = nnn
COPIES = (nnn[,(group-value[,group-value]...)])
Specifies how many copies of the sysout data set are to be printed in page sequence order, or from a 3800 Printing Subsystem, grouped by page.

If you route a job that has a COPIES parameter, the parameter will be used only if the receiving node is a JES2 node.

nnn
Specifies how many copies of the sysout data set are to be printed; each copy will be in page sequence order. The nnn is 1 through 3 decimal numbers from 1 through 255, subject to an installation-specified limit. For a data set printed on a 3800, JES2 ignores nnn if any group values are specified.

If you omit or incorrectly code the nnn parameter of COPIES, it defaults to 1 and a warning message is issued.

group-value

Specifies how many copies of each page are to be printed before the next page is printed. Each group-value is 1 through 3 decimal numbers from 1 through 255. You can code a maximum of eight group-values. Their sum must not exceed 255 or the installation-specified limit. The total copies of each page equals the sum of the group-values.

Note:

- This subparameter is valid only for 3800 output.
- For 3800 output, this subparameter overrides the nnn subparameter.

The following are not valid:

- A null group-value, for example, COPIES=(5(,)) or COPIES=(5,)
- A zero group-value, for example, COPIES=(5,(1,0,4))
- A null within a list of group-values, for example, COPIES=(5,(1,,4))

COPYG=group-value
COPYG=(group-value[,group-value]...)

Specifies how many copies of each page are to be printed before the next page is printed. Each group-value is 1 through 3 decimal numbers from 1 through 255. You can code a maximum of eight group-values. Their sum must not exceed 255. The total copies of each page equals the sum of the group-values.

Note: This parameter is valid only for 3800 output. If you code COPYG and JES2 prints the data set on an impact printer, JES2 ignores COPYG.

DEST=destination
DEST=(destination[,destination]...)

Specifies one to four different destinations for the sysout data set. The destination subparameters follow:

LOCAL

Indicates any local device.

name

Identifies a local or remote device by a symbolic name defined by the installation during JES2 initialization. The name is 1 through 8 alphanumeric or national ($, #, @) characters.

Nnnnn

Identifies a node. nnnn is 1 through 4 decimal numbers from 1 through 1000. For example, N0103.

NnnRmmmm
NnnnRmmm
NnnnnRmm

Identifies a node and a remote work station connected to the node. The node number, indicated in the format by n, is 1 through 4 decimal numbers from 1 through 1000. The remote work station number, indicated in the format by m, is 1 through 4 decimal numbers from 1 through 9999. Do not code leading zeros in n or m. The maximum number of digits for n and m combined cannot exceed six. Note that R0 is equivalent to LOCAL specified at node Nn.

nodename.userid
nodename:userid
nodename/userid
nodename(userid)

Identifies a destination node and a TSO or VM userid at that node. Use this parameter to route a sysout data set between JES2 nodes and non-JES2 nodes. The nodename is a symbolic name defined by the installation during initialization; nodename is 1 through 8 alphanumeric or national ($, #, @) characters. The userid must be defined at the node; userid for TSO is 1 through 7 alphanumeric or national ($, #, @) characters and for VM is 1 through 8 alphanumeric or national ($, #, @) characters.

Use the form **nodename.userid** to specify up to four destinations using continuation statements. The continuation statement must contain the characters /* in columns 1 and 2, OUTPUT in columns 3 through 8, a blank in column 9, an * in or following column 10, followed by one or more blanks, and the characters DEST= with the specified destinations. For example:

```
/*OUTPUT ABCD DEST=(POK.USER27,NYC.USER31)
/*OUTPUT *  DEST=(BOCA.USER58,STL.USER22)
```

Use the form **nodename:userid** to submit a job to an MVS system from a VM userid and to receive the output in the VM user's virtual reader.

Use the form **nodename.userid** to send the output to the local VM printer.

If you specify a TSO userid, do not specify a nodename that is the same as the origin node.

Note: If a data set is queued for transmission and an operator changes its destination, the userid portion of the original routing is lost.

Rnnnn
RMnnnn
RMTnnnn

Identifies a remote terminal. nnnn is 1 through 4 decimal numbers from 1 through 9999. With remote pooling, the installation may translate this route code to another route code. Note that R0 is equivalent to LOCAL.

Unnnn

Identifies a local terminal with special routing. nnnn is 1 through 4 decimal numbers from 1 through 9999.

FCB=xxxx

Identifies the forms control buffer (FCB) image JES2 is to use to guide printing of the sysout data set. The xxxx is 1 through 4 alphanumeric or national ($, #, @) characters and is the last characters of a SYS1.IMAGELIB member name:

- FCB2xxxx member, for a 3211 Printer, a 3203 Printer Model 5, or a printer supported by systems network architecture (SNA).
- FCB3xxxx member, for a 3800 Printing Subsystem.
- FCB4xxxx member, for a 4248 Printer.

IBM provides two standard FCB images. Code STD1 or STD2 only to request them.

- STD1, which specifies 6 lines per inch on an 8.5-inch-long form. (3211 and 3203-5 only)
- STD2, which specifies 6 lines per inch on an 11-inch-long form. (3211 and 3203-5 only)

If the printer on which JES2 is to print the data set does not have the forms control buffer feature, JES2 sends the operator a message to mount the proper carriage control tape.

FLASH=overlay-name
FLASH=(overlay-name[,count])
FLASH=NONE

Identifies the forms overlay to be used in printing the sysout data set on a 3800 Printing Subsystem and, optionally, specifies the number of copies on which the forms overlay is to be printed.

overlay-name

Identifies the forms overlay frame that the operator is to insert into the printer before printing begins. The name is 1 through 4 alphanumeric or national ($, #, @) characters.

Do not omit the overlay-name. The count subparameter is optional. If you omit it, you can omit the parentheses. However, if you omit it, you must not code it as a null; for example, FLASH=(ABCD,) is invalid.

Before printing starts, JES2 does not verify that the operator inserted the correct forms overlay frame for flashing.

count

Specifies the number, 1 through 255, of copies that JES2 is to flash with the overlay, beginning with the first copy printed.

JES2 determines the maximum number of copies to flash with the forms overlay by the value of nnn or the group-value total on the COPIES parameter. If the FLASH count value is greater than the value from the COPIES parameter, JES2 prints with the forms overlay the lower value.

The count subparameter of the FLASH parameter overrides the count value of the FLASHC parameter.

NONE
Suppresses flashing for this sysout data set.

Defaults: If you omit this parameter and did not specify FLASH on the DD statement or FLASHC on the /*OUTPUT statement, JES2 uses the default specified at JES2 initialization.

If you specify an overlay-name without specifying a count, JES2 flashes all copies. That is, the default for count is 255. If you specify 0 for count, JES2 also flashes all copies.

FLASHC=count
Specifies the number, 0 through 255, of copies that JES2 is to flash with the overlay, beginning with the first copy printed.

Note: For the 3800 printer, if you specify FLASH and omit FLASHC, JES2 flashes all copies.

The count subparameter of the FLASH parameter overrides the count value of the FLASHC parameter.

FORMS=xxxx
FORMS=STD
Identifies the forms on which JES2 is to print or punch the sysout data set.

xxxx
Identifies the print or punch forms. form-name is 1 through 4 alphanumeric or national ($, #, @) characters.

STD
Indicates that JES2 is to use the default specified at JES2 initialization.

INDEX=nn
Sets the left margin for output on a 3211 Printer with the indexing feature. The width of the print line is reduced by the INDEX parameter value. The nn specifies how many print positions the left margin on the 3211 output is to be indented. nn is a decimal number from 1 through 31. n=1 indicates flush-left; n=2 through n=31 indent the print line by n-1 positions.

JES2 ignores the INDEX parameter if the printer is not a 3211 with the indexing feature.

INDEX and LINDEX are mutually exclusive; if you code both, JES2 uses the last one encountered.

LINDEX=nn
Sets the right margin for output on a 3211 Printer with the indexing feature. The width of the print line is reduced by the LINDEX parameter value. The nn specifies how many print positions the right margin on 3211 output is to be moved in from the full page width. nn is a decimal number from 1 through 31. n=1 indicates flush-right; n=2 through n=31 move the right margin over by n-1 positions.

JES2 ignores the LINDEX parameter on all printers except the 3211 with the indexing feature.

INDEX and LINDEX are mutually exclusive; if you code both, JES2 uses the last one encountered.

LINECT = nnn

Specifies the maximum number of lines JES2 is to print on each output page. The nnn is a number from 0 through 255.

Specify LINECT = 0 to keep JES2 from starting a new page when the number of lines exceeds the JES2 initialization parameter.

If you code LINECT on the /*OUTPUT statement, it overrides the LINECT value on the /*JOBPARM statement and the linect value in the accounting information parameter of the JOB statement.

If the LINECT parameter is omitted from the /*OUTPUT statement, JES2 obtains the value from one of the following sources, in order:

1. The LINECT parameter on the /*JOBPARM statement.
2. The linect field of the accounting information parameter on the JOB statement.
3. The installation default specified at JES2 initialization.

MODIFY = module-name
MODIFY = (module-name[,trc])

Specifies a copy-modification module that tells JES2 how to print the sysout data set on a 3800 Printing Subsystem. The module can specify legends, column headings, blanks, and where and on which copies the data is to be printed. The module is defined and stored in SYS1.IMAGELIB using the IEBIMAGE utility program.

module-name

Identifies a copy-modification module in SYS1.IMAGELIB. The module-name is 1 through 4 alphanumeric or national ($, #, @) characters.

Do not omit the module-name.

trc

Identifies which table-name in the CHARS parameter is to be used. This **table reference character** is 0 for the first table-name specified, 1 for the second, 2 for the third, or 3 for the fourth.

If you do not specify **trc**, the default is 0. If the trc value is greater than the number of table-names in the CHARS parameter, JES2 uses the first table named in the CHARS parameter.

The trc subparameter is optional. If you omit it, you can omit the parentheses. However, if you omit it, you must not code it as a null; for example, MODIFY = (TAB1,) is invalid. If you omit the trc subparameter, JES2 uses the first table-name.

The trc subparameter of the MODIFY parameter overrides the trc subparameter of the MODTRC parameter.

MODTRC = trc

Identifies which table-name in the CHARS parameter is to be used. This **table reference character** is 0 for the first table-name specified, 1 for the second, 2 for the third, or 3 for the fourth.

If you do not specify **trc**, the default is 0. If the trc value is greater than the number of table-names in the CHARS parameter, JES2 uses the first table named in the CHARS parameter.

The trc subparameter of the MODIFY parameter overrides the trc subparameter of the MODTRC parameter.

UCS=xxxx
Identifies the universal character set (UCS) image JES2 is to use in printing the sysout data set. The xxxx is 1 through 4 alphanumeric or national ($, #, @) characters. See Figure 10-2 on page 10-171 for IBM standard special character set codes.

Overrides

- /*OUTPUT statement parameters override all equivalent DD statement parameters.
- If a /*OUTPUT statement contains duplicate parameters, the last parameter overrides all preceding duplicates, except for the DEST parameter.
- Any parameter coded on subsequent /*OUTPUT statements overrides the same parameter on previous /*OUTPUT statements.
- JES2 adds any parameter you code on subsequent /*OUTPUT statements that you did not code on previous /*OUTPUT statements to the previous /*OUTPUT statement.
- If you code LINECT on the /*OUTPUT statement, it overrides the LINECT value on the /*JOBPARM statement and the linect value in the accounting information parameter of the JOB statement.

Relationship to Other Control Statements

- JES2 processes /*OUTPUT statements placed after a /*ROUTE XEQ statement at the execution node only.
- JES2 processes /*OUTPUT statements placed before a /*ROUTE XEQ statement at both the input node and the execution node.

Location in the JCL

Place the /*OUTPUT statement after the JOB statement and before the first EXEC statement.

Example of the /*OUTPUT Statement

```
/*OUTPUT ABCD COPIES=6,COPYG=(1,2,3),DEST=RMT23
```

This statement refers to all sysout data sets defined by a DD statement that specifies SYSOUT=(c,,ABCD). Six copies of each page of output are printed. If the printer is a 3800, first one copy of each page is printed, then two copies of each page, and finally, three copies of each page. If the printer is not a 3800, COPYG is ignored and six copies of the entire data set are printed. The output is sent to remote terminal 23.

/*PRIORITY Statement

Purpose: Use the /*PRIORITY statement to assign a selection priority for your job and all of its output, except the JES2 hard-copy log. Within a job class, a job with a higher priority is selected for execution sooner.

Note: Depending on the JES2 initialization options in use at your installation, JES2 may ignore the /*PRIORITY statement.

Syntax:

```
/*PRIORITY p
```

The /*PRIORITY statement consists of the characters /* in columns 1 and 2, PRIORITY in columns 3 through 10, a blank in column 10, and the priority starting in any column from 11 through 71. JES2 ignores columns 72 through 80.

Parameter Definition

p

Requests a priority. The p is 1 or 2 decimal numbers from 0 through 15. The highest priority is 15.

Follow your installation's rules in coding a priority.

Overrides

A priority specified on a /*PRIORITY statement overrides a priority specified in the PRTY parameter on a JOB statement.

Relationship to Other Control Statements

The system derives the priority from the following, in override order:

1. JES2 /*PRIORITY statement.
2. The PRTY parameter on the JOB statement.
3. The accounting information on a /*JOBPARM statement.
4. The accounting information on the JOB statement.
5. An installation default specified at JES2 initialization.

JES2: /*PRIORITY

Location in the JCL

The /*PRIORITY statement must immediately precede the JOB statement. If not, or if **p** is not a number from 0 through 15, JES2 ignores the /*PRIORITY statement and flushes the input stream until the next JOB statement or another /*PRIORITY statement.

In a JES2 network, the /*PRIORITY statement must immediately follow the /*XMIT statement and precede a JOB statement. If the /*PRIORITY statement does not immediately follow an /*XMIT statement, JES2 ignores the /*PRIORITY statement at any node except the input node.

Example of the PRIORITY Statement

```
/*PRIORITY 7
```

This statement assigns a job queue selection priority of 7. This value has meaning only in relation to other jobs in the system.

/*ROUTE Statement

Purpose: Use the /*ROUTE statement to specify the destination of sysout data sets that are not routed by a DEST parameter or to identify the network node where the job is to execute.

Syntax:

```
                                  ( LOCAL                )
                                  ( name                 )
                                  ( Nnnnn                )
                                  ( NnnRmmmm             )
                                  ( NnnnRmmm             )
                                  ( NnnnnRmm             )
/*ROUTE  { PRINT } ( nodename.userid       )
          { PUNCH } ( nodename:userid      )
                    ( nodename/userid      )
                    ( nodename(userid)     )
                    ( Rnnnn                )
                    ( RMnnnn               )
                    ( RMTnnnn              )
                    ( Unnnn                )

                    ( Nnnnn                )
/*ROUTE XEQ         ( nodename.vmguestid   )
                    ( nodename:vmguestid   )
                    ( nodename/vmguestid   )
                    ( nodename(vmguestid)  )
```

The /*ROUTE statement consists of the characters /* in columns 1 and 2; ROUTE in columns 3 through 7; at least one blank followed by PRINT, PUNCH, or XEQ; at least one blank followed by one of the destinations or nodes; and at least one blank before column 72. JES2 ignores columns 72 through 80.

Code only one destination or node on each /*ROUTE statement.

Parameter Definition

PRINT
Requests that JES2 route the job's sysout data sets that are printed.

PUNCH
Requests that JES2 route the job's sysout data sets that are punched.

XEQ
Requests that JES2 route the job to a network node for execution.

LOCAL
Indicates any local device at the node at which the job is submitted.

name
Identifies a local or remote device by a symbolic name defined by the installation during JES2 initialization. The name is 1 through 8 alphanumeric or national ($, #, @) characters.

Nnnnn
Identifies a node. nnnn is 1 through 4 decimal numbers from 1 through 1000. For example, N0103.

NnnRmmmm
NnnnRmmm
NnnnnRmm
Identifies a node and a remote work station connected to the node. The node number, indicated in the format by n, is 1 through 4 decimal numbers from 1 through 1000. The remote work station number, indicated in the format by m, is 1 through 4 decimal numbers from 1 through 9999. Do not code leading zeros in n or m. The maximum number of digits for n and m combined cannot exceed six.

Note: R0 is equivalent to specifying LOCAL at node Nn.

nodename.userid
nodename:userid
nodename/userid
nodename(userid)
Identifies a node and a VM or TSO userid, a remote workstation, or a symbolic name defined at the destination node. The node is a symbolic name defined by the installation during initialization; nodename is 1 through 8 alphanumeric or national ($, #, @) characters. The userid must be defined at the node; userid is 1 through 8 alphanumeric or national ($, #, @) characters.

A userid requires a node; therefore, code nodename.userid. You **cannot** code a userid without a nodename.

If you specify a TSO userid, do not specify a nodename that is the same as the origin node.

Note: If a data set is queued for transmission and an operator changes its destination, the userid portion of the routing is lost.

Rnnnn
RMnnnn
RMTnnnn
Identifies a remote terminal. nnnn is 1 through 4 decimal numbers from 1 through 9999. Note that with remote pooling, the installation may translate this route code to another route code.

Note: R0 is equivalent to LOCAL.

Unnnn
Identifies a local terminal with special routing. nnnn is 1 through 4 decimal numbers from 1 through 9999.

nodename.vmguestid
nodename:vmguestid
nodename/vmguestid
nodename(vmguestid)

Identifies the network node where the job is to execute. The nodename identifies an MVS JES2 system, an MVS JES3 system, a VSE POWER node, or a VM system. If nodename specifies the local node, the job executes locally. The nodename is 1 through 8 alphanumeric, national ($, #, @), or special characters specified during JES2 initialization.

The vmguestid identifies a guest system running in a virtual machine (VM), for example, an MVS system running under VM. Do not specify a work station or terminal in this parameter.

Location in the JCL

Place the /*ROUTE statement after the JOB statement and either before or after the EXEC statements. Place a /*ROUTE XEQ statement before all DD * or DD DATA statements in the job.

Processing of /*ROUTE Statements

- If you do not specify a node on the /*ROUTE PRINT or PUNCH statement, printing or punching occurs at the input node.
- JES2 processes /*ROUTE XEQ statements on the input node only.
- When a /*ROUTE PRINT or PUNCH statement follows a /*ROUTE XEQ statement, JES2 processes the /*ROUTE PRINT or PUNCH statement on the execution node only. However, printing or punching occurs at the node specified on the /*ROUTE PRINT or PUNCH statement.
- When a /*ROUTE PRINT or PUNCH statement precedes a /*ROUTE XEQ statement, JES2 processes the /*ROUTE PRINT or PUNCH statement on both the input and execution nodes. However, printing or punching occurs at the node specified on the /*ROUTE PRINT or PUNCH statement.

Multiple /*ROUTE Statements

JES2 uses the last /*ROUTE statement of each category, if a job contains more than one /*ROUTE PRINT or PUNCH or XEQ statement.

Examples of the ROUTE Statement

```
/*ROUTE  PRINT RMT6
```

This statement sends the printed output to remote terminal 6.

```
/*ROUTE  PUNCH PUN2
```

This statement sends the punched output to device PUN2, which was identified to the system during initialization.

```
//JOBB   JOB    ...
/*ROUTE  XEQ    DENVER
//STEP1  EXEC   ...
          .
          .
```

This statement sends the job to the node named DENVER for execution. The entire job is scanned for JCL errors on the input system before being transmitted to the target system. The entire job is transmitted, which includes the JOBB JOB statement. Options on the JOBB JOB statement apply to both the input and target system.

/*SETUP Statement

Purpose: Use the /*SETUP statement to identify volumes that the operator should mount before the job is executed. When the job enters the system, JES2 issues a message to the operator console, asking the operator to mount the identified volumes. JES2 then places the job in hold status until the operator mounts the volumes and releases the job.

Syntax:

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

The /*SETUP statement consists of the characters /* in columns 1 and 2, SETUP in columns 3 through 7, a blank in column 10, and the volume serial number(s) starting in any column from 11 through 71. JES2 ignores columns 72 through 80.

Do not continue the /*SETUP statement; code as many /*SETUP statements as necessary.

Parameter Definition

serial-number
Identifies by serial number the volume(s). A volume serial number is 1 through 6 alphanumeric, national ($, #, @), or special characters; enclose a serial number that contains special characters, other than hyphens, in apostrophes. If the number is shorter than 6 characters, it is padded with trailing blanks.

Location in the JCL

Place all /*SETUP statements after the JOB statement and before the first EXEC statement.

To prevent JES2 from requesting mounting of volumes on a node other than the node of execution, the /*SETUP statement should follow any /*ROUTE XEQ or /*XEQ statement. If JES2 processes the /*SETUP statement before processing a /*ROUTE XEQ or /*XEQ statement, JES2 requests the setup on both the input and execution nodes.

Example of the /*SETUP Statement

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

This statement requests that volumes 666321 and 149658 be mounted for the job.

/*SIGNOFF Statement

Purpose: Use the /*SIGNOFF statement to tell JES2 to end a remote job stream processing session. At the completion of the current print and/or punch streams, JES2 disconnects the remote work station from the system. If JES2 is reading jobs from the station when the output completes, JES2 disconnects the remote station when the input is completed.

Note: The remote terminal access processor processes the /*SIGNOFF statement if it appears in a job stream.

Both systems network architecture (SNA) and binary synchronous communication (BSC) remote work stations can use the /*SIGNOFF statement. SNA remote stations can also use the LOGOFF command to end a session with JES2. The LOGOFF command has some options that the /*SIGNOFF statement does not provide.

References: For information on the LOGOFF command, see *SPL: JES2 Initialization and Tuning* and *Advanced Communications Function for VTAM Version 2 Programming.*

Syntax:

```
/*SIGNOFF
```

The /*SIGNOFF statement consists of the characters /* in columns 1 and 2, SIGNOFF in columns 3 through 9, and blanks in columns 10 through 80.

Location in the JCL

The /*SIGNOFF statement can appear anywhere in a local input stream or an input stream from a SNA or BSC remote work station.

Example of the /*SIGNOFF Statement

```
/*SIGNOFF
```

This statement requests that JES2 terminate a remote job stream processing session.

/*SIGNON Statement

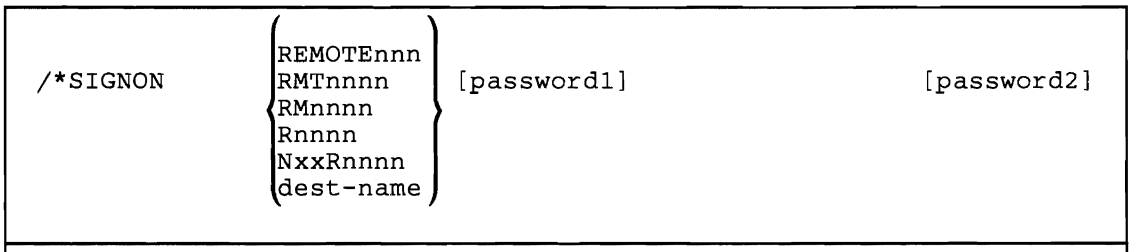
Purpose: Use the /*SIGNON statement to tell JES2 to begin a remote job stream processing session. For non-multi-leaving remote stations, the terminal transmits the /*SIGNON statement alone as part of the initial connection process.

Note: The remote terminal access processor processes the /*SIGNON statement if it appears in a job stream. When the terminal access processor processes the /*SIGNON statement, the line being processed is restarted.

Systems network architecture (SNA) remote work stations must use the LOGON command instead of the /*SIGNON statement to notify JES2 of a connection request.

References: For information on the LOGON command, see *SPL: JES2 Initialization and Tuning* and *ACF/VTAM Version 2 Programming*.

Syntax:



The /*SIGNON statement consists of the following. Note that all the fields in this statement must appear in fixed locations.

Column	Contents
1-8	/*SIGNON
16-23	REMOTEnnn, RMTnnnn, RMnnnn, Rnnnn, NxxRnnnn, or dest-name beginning in 16
25-32	password1, beginning in 25
73-80	password2, beginning in 73

Parameter Definition

REMOTEnnn
Specifies the identification number assigned to the remote station asking to sign on. nn is 1 through 3 decimal numbers.

Code REMOTEnnn with the same characters as RMTnnn on the /*ROUTE statement. If you code REMOTEnnn on the /*SIGNON statement, you are restricted to coding RMTnnn with only three numbers on the /*ROUTE statement.

RMTnnnn
RMnnnn
Rnnnn
Specifies the identification number assigned to the remote station. nnnn is 1 through 4 decimal numbers.

NxxRnnnn
Specifies the node number in the NJE network and the identification number assigned to the remote station. Nxx must specify the node to which the remote station is connected. xx is 1 through 1000. nnnn is 1 through 4 decimal numbers. xx plus nnnn cannot exceed 6 numbers.

dest-name
Specifies the name (1 through 8 characters) that you use to refer to the JES2-defined destination. The dest-name must be defined as a remote station on the system to which the terminal is connected.

password1
Specifies the password assigned to a non-dedicated connection that allows the remote station access to JES2 for remote job stream processing. The installation assigns this password during JES2 initialization. The operator can change or delete this password with the $T command.

password2
Specifies the password for the remote station that is signing on; this password identifies the remote station as a valid remote job entry (RJE) station. The installation assigns this password during JES2 initialization.

Location in the JCL

Place the /*SIGNON statement at the start of an input stream to be transmitted from a remote work station. For non-multi-leaving remote stations, the terminal transmits the /*SIGNON statement alone as part of the initial connection process.

Place the /*SIGNON statement at the end of the JES2/RTP input stream for multi-leaving remote stations.

Examples of the /*SIGNON Statement

```
/*SIGNON       REMOTE123LINEPSWD
```

This statement requests that remote station 123 begin a remote job stream processing session. LINEPSWD, beginning in column 25, is the password assigned to the non-dedicated connection.

```
/*SIGNON       RMT1000  LINEPSWD                                    PSWD2
```

This statement requests that remote station 1000 begin a remote job stream processing session. LINEPSWD, beginning in column 25, is the password assigned to the non-dedicated connection. PSWD2, beginning in column 73, is the password assigned to the remote station 1000.

```
/*SIGNON       N11R123  LINEPSWD
```

This statement requests that remote station 123 at node 11 begin a remote job stream processing session. LINEPSWD, beginning in column 25, is the password assigned to the non-dedicated connection.

/*XEQ Statement

Purpose: Use the /*XEQ statement to identify the network node where the job is to execute. It performs the same function as the /*ROUTE XEQ statement.

Syntax:

```
/*XEQ {Nnnnn                 }
     {nodename[.vmguestid]}
```

The /*XEQ statement consists of the characters /* in columns 1 and 2, XEQ in columns 3 through 5, a blank in column 6, and a node starting in any column starting with 7.

Parameter Definition

Nnnnn
Identifies a node. nnnn is 1 through 4 decimal numbers from 1 through 1000. For example, N0103.

nodename
Identifies the network node where the job is to execute. The nodename identifies an MVS JES2 system, an MVS JES3 system, a VSE POWER node, or a VM system. If nodename specifies the local node, the job executes locally. The nodename is 1 through 8 alphanumeric, national ($, #, @), or special characters specified during JES2 initialization.

vmguestid
Identifies a guest system running in a virtual machine (VM), for example, an MVS system running under VM. Do not specify a work station or terminal in this parameter.

Location in the JCL

Place the /*XEQ statement after the JOB statement and either before or after the EXEC statements. Place a /*XEQ statement before all DD * or DD DATA statements in the job.

Multiple /*XEQ Statements

JES2 uses the last /*XEQ statement, if a job contains more than one /*XEQ statement.

Example of the XEQ Statement

```
//JOBB    JOB  ...
/*XEQ     ATLANTA
//STEP1   EXEC  ...
           .
           .
```

JES2 routes and executes this job on the node defined as ATLANTA. The entire job is scanned for JCL errors on the input system before being transmitted to the target system. The entire job is transmitted, which includes the JOBB JOB statement. Options on the JOBB JOB statement apply to both the input and target system.

/*XMIT Statement

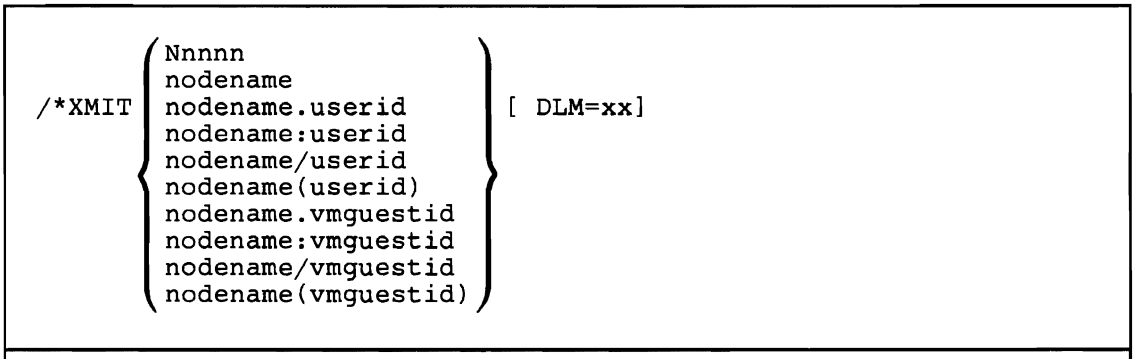
Purpose: Use the /*XMIT statement to transmit records from a JES2 node to either another JES2 node or an eligible non-JES2 node, for example, a VM or JES3 node. JES2 does not process or check the records for JES2 validity. JES2 builds header and trailer records from information on the JOB statement immediately preceding the /*XMIT statement. Then JES2 transmits all the records following the /*XMIT statement.

The records may consist of a job input stream or an in-stream DD * or DD DATA data set. If the records are a job input stream and the destination node can process JCL, the transmitted input stream is executed; in this case, the record immediately following the /*XMIT statement must be a JOB statement that is valid at the destination node.

The records end when JES2 finds one of the following:

/* in the input stream
The two-character delimiter specified by a DLM parameter on this /*XMIT statement
The input stream runs out of card images
If the records are being read from an internal reader, the internal reader is closed

Syntax:



The /*XMIT statement consists of the characters /* in columns 1 and 2, XMIT in columns 3 through 6, a blank in column 7, a nodename or node-number starting in any column starting with 8, and optionally followed, with an intervening blank, by a delimiter parameter.

Do not continue an /*XMIT statement.

Parameter Definition

Nnnnn
Identifies the destination node. nnnn is 1 through 4 decimal numbers from 1 through 1000. For example, N0103.

nodename
Identifies the destination node. The nodename identifies an MVS JES2 system, an MVS JES3 system, a VSE POWER node, or a VM system. The nodename is 1 through 8 alphanumeric, national ($, #, @), or special characters specified during JES2 initialization.

userid
Identifies a destination terminal or work station at the node. The userid must be defined at the node; userid for TSO is 1 through 7 alphanumeric or national ($, #, @) characters and for VM is 1 through 8 alphanumeric or national ($, #, @) characters.

Note: If the data set is queued for transmission and an operator changes its destination, the userid portion of the original routing is lost.

vmguestid
Identifies a destination guest system running in a virtual machine (VM), for example, an MVS system running under VM. Do not specify a work station or terminal in this parameter.

Note: If the data set is queued for transmission and an operator changes its destination, the vmguestid portion of the original routing is lost.

DLM=xx
Specifies a two-character delimiter to terminate the data being transmitted.

Code any two characters for the delimiter. If the specified delimiter contains any special characters, enclose it in apostrophes. In this case, a special character is any character that is neither alphanumeric nor national ($, #, @).

Failing to code enclosing apostrophes produces unpredictable results.

If the delimiter contains an ampersand or an apostrophe, code each ampersand or apostrophe as two consecutive ampersands or apostrophes. Each pair of consecutive ampersands or apostrophes counts as one character.

If you specify a DLM parameter, you must terminate the transmitted records with the characters in the DLM parameter. The characters you assign as delimiters override any delimiter implied by the defaults.

The characters // are not valid delimiters unless specifically indicated by DLM=//.

Defaults

For the end of the records to be transmitted, the default is /* in the input stream.

If you specify for DLM only one character or more than two characters, JES2 uses /*.

Location in the JCL

Place the /*XMIT statement immediately after a JOB statement.

Code only one /*XMIT statement in a job.

Examples of the XMIT Statement

```
//JOBA  JOB  ...
/*XMIT  ATLANTA DLM=AA
         .
         records to be transmitted
         .
AA
```

JES2 transmits to the node ATLANTA all records following the /*XMIT statement up to the specified delimiter, AA.

```
//JOBX  JOB  ...
/*XMIT  VMSYS1.MVS223
//JOBB  JOB  ...
         .
         job to be transmitted
         .
/*
```

JES2 transmits the JOBB job stream to the VM guest system, MVS223, running on node VMSYS1, which is a VM system. The job stream will be executed by the MVS223 system.

The information specified on the JOBX statement is processed on the submitting system and transmitted in the networking headers to the target system. (If the target system is a JES2 node, this header information is not used.)

Chapter 22. JES3 Control Statements

Code JES3 control statements with JCL statements to control the input and output processing of jobs. The rules for coding in Chapter 3, "Format of Statements" and Chapter 4, "Syntax of Parameters" apply to the JES3 control statements.

Location in the JCL

Place JES3 control statements, except the command and //*PAUSE statements, after the JOB statement and its continuations. JES3 ignores JES3 control statements, except the command and //*PAUSE statements, that appear before the JOB statement or between continued JOB statements.

Do not include JES3 control statements in a cataloged or in-stream procedure. JES3 ignores JES3 control statements in a procedure.

Internal Reader

Use the following control statements when submitting jobs to the internal reader. The internal reader is described in *SPL: JES3 Initialization and Tuning*, and *SPL: System Modifications.*

```
/*DEL
/*EOF
```

Examples of JES3 Control Statements

The following shows JES3 control statements in relation to each other and to JCL statements for a job entered from a remote work station. No actual job should require all of these statements.

```
//**MESSAGE,CN1,ENTER A START COMMAND FOR THIS JOB
//*PAUSE
//TEST1    JOB    ,,MSGCLASS=A
//*NETACCT PNAME=MAEBIRD,ACCT=2K14920
//*NET NETID=N1,NHOLD=0
//*PROCESS CI
//*PROCESS MAIN
//*PROCESS OUTSERV
//*DATASET DDNAME=STEP1.DD1
         .
         .
         data
         .
//*ENDDATASET
//*ENDPROCESS
//*OPERATOR THIS IS TEST JOB TEST1.
//*MAIN CLASS=C
//*FORMAT PR,DDNAME=STEP1.DD2,DEST=ANYLOCAL,COPIES=2
//*ROUTE XEQ NODE1
//FARJOB1  JOB    ,,MSGCLASS=A
//STEP1    EXEC   PGM=CHECKER
//DD1      DD     DSNAME=INPUT
//DD2      DD     SYSOUT=A
/*
```

The following is an ordinary job entered through the local input stream.

```
//RUN2     JOB    ,,MSGCLASS=A
//*MAIN CLASS=B
//*FORMAT PR,DDNAME=STEPA.DD2,DEST=ANYLOCAL,COPIES=5
//STEPA    EXEC   PGM=WRITER
//DD1      DD     DSNAME=IN1,DISP=OLD,UNIT=3350,VOLUME=SER=MH2244
//DD2      DD     SYSOUT=A
/*
```

JES3 Command Statement

Purpose: Use the command statement to enter a JES3 operator command through the input stream.

JES3 usually executes an in-stream command as soon as it is read. Therefore, the command will **not** be synchronized with the execution of any job or step in the input stream. To synchronize a command with job processing, tell the operator the commands you want and when they should be issued, then let the operator enter them from the console.

References: For more information on the command statement and the JES3 verbs and operands, see *JES3 Commands*.

Syntax:

```
//**command-verb[,operand]...
```

The JES3 command statement consists of the characters //** in columns 1 through 4, the command verb beginning in column 5, and, if the command requires operands, a comma followed by the operands up through column 72. JES3 ignores columns 73 through 80.

Do not continue command statements from one statement to the next.

Parameter Definition

***command-verb**

Indicates one of the following JES3 commands. **Do not** specify a *DUMP or *RETURN command on a JES3 command statement.

Command	Short Form
CALL	X
CANCEL	C
DELAY	D
DISABLE	H
ENABLE	N
ERASE	E
FAIL	
FREE	
INQUIRY	I
MESSAGE	Z
MODIFY	F
RESTART	R
SEND	T
START	S
SWITCH	
VARY	V

operand

Specifies an operand that pertains to the command-verb.

JES3: Command

Location in the JCL

- Place JES3 command statements before the first JOB statement in the input stream, if jobs are also being submitted. JES3 treats any JES3 command statements that follow the JOB statement as comment statements.
- You can enter several command statements at one time.
- You can place command statements at the beginning of the cards in an active card reader that is being restarted.
- Command statements can be entered through card, tape, or disk readers.
- Command statements **cannot** be entered through an internal reader or from another node.

Examples of the Command Statement

```
//**VARY,280,OFFLINE
//**V,281,OFFLINE
//**VARY,282,OFF

//**V,280-282,OFF
```

In this example, the first three statements each vary one device offline. Alternatively, the fourth statement varies all three devices offline. If you place these cards in card reader 01C, for example, and it is currently not in use, the operator would enter through the operator console:

```
*X CR,IN=01C
```

```
//**MESSAGE,CN1,OUTPUT FROM JOB X REQUIRES SPECIAL CONTROLS
```

This statement instructs the operator from a remote location. Place this statement before the first job in the input stream.

//*DATASET Statement

Purpose: Use the //*DATASET statement to identify the beginning of an in-stream data set, which can contain JCL and/or data. (The //*ENDDATASET statement ends the in-stream data set.) The data set can be used as input to a dynamic support program (DSP), such as OUTSERV.

Note: Make sure the operator includes a **C** operand on the *CALL command for the reader that reads a job containing this statement if it contains a MODE=C parameter.

Syntax:

```
//*DATASET DDNAME=ddname[,parameter]...
The parameters are:
     MODE={E|C}
     J={YES|NO}
     CLASS={NO|MSGCLASS|class}
```

The //*DATASET statement consists of the characters //* in columns 1 through 3, DATASET in columns 4 through 10, a blank in column 11, and parameters in columns 12 through 72. JES3 ignores columns 73 through 80.

Parameter Definition

DDNAME=ddname
Specifies the name of the in-stream data set that follows the //*DATASET statement.

MODE=E
MODE=C
Defines the card-reading mode.

E
Indicates that JES3 is to read the cards as EBCDIC with validity checking. E is the default if the MODE parameter is omitted.

C
Indicates that JES3 is to read the cards in card image form, that is, in column binary or data mode 2.

MODE=C is not valid for jobs read from disk or tape, or for jobs submitted from remote work stations.

J=YES
J=NO
Indicates how JES3 is to recognize the end of the data set.

If you specify MODE=C, JES3 ignores the J parameter; therefore, use a //*ENDDATASET statement to end the data set.

YES
Indicates that a //*ENDDATASET statement ends the data set. Specify YES when JOB statements appear in the data set.

NO
Indicates that a JOB statement ends the data set. NO is the default if the J parameter is omitted, unless MODE=C is specified.

CLASS=NO
CLASS=MSGCLASS
CLASS=class
Identifies the output class JES3 is to use for the data set.

NO
Indicates that the system is to assign an output class. If you omit the CLASS parameter, the default is NO.

MSGCLASS
Requests the output class in the MSGCLASS parameter on the JOB statement.

class
Specifies the output class.

Location in the JCL

Place a //*DATASET statement immediately before the first record of an in-stream data set.

Example of the //*DATASET Statement

```
//*PROCESS OUTSERV
//*DATASET DDNAME=MYPRINT,J=YES
           .
           .
         data
           .
           .
//*ENDDATASET
//*FORMAT PR,DDNAME=MYPRINT,COPIES=5
//STEP1   EXEC ...
           .
           .
```

In this example, the //*DATASET statement marks the beginning of the in-stream data set MYPRINT. The //*FORMAT PR statement requests five copies of it. A //*ENDDATASET statement marks the end of the data set.

//*ENDDATASET Statement

Purpose: Use the //*ENDDATASET statement to indicate the end of an in-stream data set that was begun with a //*DATASET statement.

Syntax:

```
//*ENDDATASET
```

The //*ENDDATASET statement consists of the characters //* in columns 1 through 3 and ENDDATASET in columns 4 through 13. Columns 14 through 80 must be blank.

Location in the JCL

Place a //*ENDDATASET statement immediately after the last record of an in-stream data set that was begun with a //*DATASET statement.

Example of the //*ENDDATASET Statement

```
//*DATASET DDNAME=INFO,J=YES
           .
           .
           data
           .
           .
//*ENDDATASET
```

In this example, the //*ENDDATASET statement marks the end of the in-stream data set INFO.

//*ENDPROCESS Statement

Purpose: Use the //*ENDPROCESS statement to indicate the end of a series of //*PROCESS statements in a job.

Syntax:

```
//*ENDPROCESS [comments]
```

The //*ENDPROCESS statement consists of the characters //* in columns 1 through 3, ENDPROCESS in columns 4 through 13, a blank in column 14, and, optionally, comments starting in any column beginning with 15. JES3 ignores columns 73 through 80.

Location in the JCL

Place a //*ENDPROCESS statement immediately after the last //*PROCESS statement in a job. The //*ENDPROCESS statement is optional if a JCL statement follows the last //*PROCESS statement.

Do not place any //*PROCESS statements after the //*ENDPROCESS statement.

Example of the //*ENDPROCESS Statement

```
//*ENDPROCESS   END OF PROCESS STATEMENTS
```

//*FORMAT PR Statement

Purpose: Use the //*FORMAT PR statement to specify to JES3 processing instructions for sysout data sets that are printed. These instructions permit special processing of sysout data sets, such as:

- Multiple destinations.
- Multiple copies of output with different attributes.
- Forced single or double space control.
- Printer overflow checking.

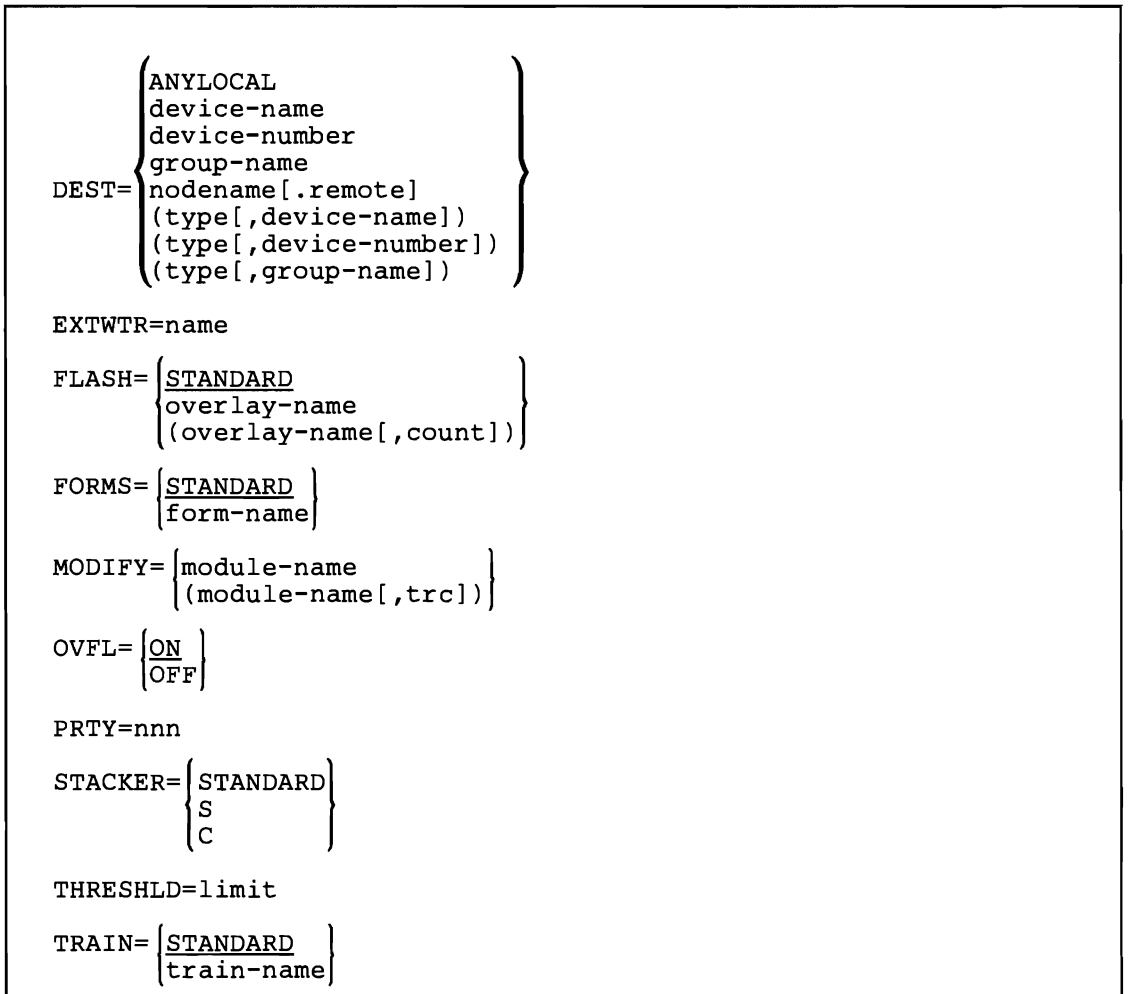
You can code several //*FORMAT PR statements for one sysout data set to specify special requirements for different copies of the data set. You can also code a //*FORMAT PU statement for the same sysout data set, thereby both printing and punching it.

Note: The //*FORMAT PR statement applies only to sysout data sets printed by JES3. The statement is ignored for data sets sent to a TSO userid or processed by an external writer.

References: For additional information, refer to the output services topic in *SPL: JES3 Intitialization and Tuning*.

Syntax:

```
//*FORMAT  PR,DDNAME={stepname.ddname                }[,parameter]...
                     {stepname.procstepname.ddname  }
                     {SYSMSG                        }
                     {JESJCL                        }
                     {JESMSG                        }

//*FORMAT PR,DDNAME=[,parameter]...

The parameters are:

 {CARRIAGE={carriage-tape-name}}
 {         {6                 }}
 {FCB={image-name}             }
 {    {6         }             }

 CHARS={STANDARD                    }
       {table-name                  }
       {(table-name[,table-name]...)}

 CHNSIZE={DS       }
         {(nnn[,mmm])}

 COMPACT=compaction-table-name

 CONTROL={PROGRAM}
         {SINGLE }
         {DOUBLE }
         {TRIPLE }

 COPIES={nnn                                  }
        {(nnn,(group-value[,group-value]...)) }
        {(group-value[,group-value]...)       }
```



The //*FORMAT PR statement consists of the characters //* in columns 1 through 3, FORMAT in columns 4 through 9, a blank in column 10, PR in columns 11 and 12, a comma in column 13, and parameters in columns 14 through 72. JES3 ignores columns 73 through 80.

Parameter Definition

PR

Indicates that this statement is associated with a sysout data set that is printed.

JES3: //*FORMAT PR

DDNAME=
DDNAME=stepname.ddname
DDNAME=stepname.procstepname.ddname
DDNAME=SYSMSG
DDNAME=JESJCL
DDNAME=JESMSG

(null)
Specifies that the parameters on this //*FORMAT PR statement are the defaults for the job. These parameters then apply to all of the job's sysout data sets that are printed, except those covered by a //*FORMAT PR statement with DDNAME=ddname.

Nonspecific and Specific Statements: A //*FORMAT PR statement that contains DDNAME= is called **nonspecific**; a statement that contains DDNAME=ddname is called **specific**.

Overrides: Parameters coded on a nonspecific //*FORMAT PR statement are overridden by parameters coded on sysout DD statements or by parameters in the JES3 SYSOUT initialization statement.

stepname.ddname
stepname.procstepname.ddname
Identifies the DD statement that defines the sysout data set to be printed. Use form **stepname.ddname** to indicate DD statement, ddname, in step, stepname, in this job. Use form **stepname.procstepname.ddname** to indicate DD statement, ddname, in procedure step, procstepname, of a procedure that is called by a step, stepname, in this job. The ddname must match exactly the ddname on the DD statement. (See the example for the //*DATASET statement.) If the identified DD statement does not contain a SYSOUT parameter, JES3 ignores the //*FORMAT PR statement.

SYSMSG
Requests printing of system messages.

JESJCL
Requests printing of JCL statements and messages.

JESMSG
Requests printing of JES3 and operator messages about the job.

CARRIAGE=carriage-tape-name
CARRIAGE=6
Specifies the carriage tape for the 3211, 3203 Model 5, or 1403 Printer for printing this output class.

carriage-tape-name
Identifies the name of the carriage tape. The name is 1 through 8 characters.

For the 3211 and 3203-5, SYS1.IMAGELIB must contain a module for each carriage tape name.

6
Indicates the installation standard carriage tape.

Note: You cannot code both the CARRIAGE and FCB parameters on the same //*FORMAT PR statement.

CHARS=STANDARD
CHARS=table-name
CHARS=(table-name[,table-name]...)

Requests one or more character-arrangement tables for printing the sysout data set on a 3800 Printing Subsystem.

STANDARD

Indicates the standard character-arrangement table, which was specified at JES3 initialization.

table-name

Identifies a character-arrangement table. Each table-name is 1 through 4 alphanumeric or national ($, #, @) characters. When coding more than one table-name, parentheses are required around the list and null positions are invalid in the list.

CHNSIZE=DS
CHNSIZE=(nnn[,mmm])

Gives the number of logical records to be transmitted to a work station as a systems network architecture (SNA) chain and indicates whether normal output checkpoints are to be taken for this sysout data set.

Note: This parameter is valid only when transmitting to a SNA work station.

Be careful in selecting subparameters, because each affects performance differently. Sending the data set as a SNA chain provides the best performance, but can cause duplicate data to be written to the output device if operator intervention is required. The remote operator can eliminate duplicate data by issuing commands to reposition and restart the output writers.

When an end-of-chain indicator is sent in the data set, JES3 takes an output checkpoint. You can provide additional checkpoints for critical data by sending an end-of-chain indicator. For example, when printing bank checks, you can have an output checkpoint taken for each check by specifying each check as a SNA chain.

DS

Indicates that the sysout data set is to be sent as a single SNA chain and that JES3 is to take normal output checkpoints. DS is the default if the CHNSIZE parameter is omitted.

nnn

Specifies the SNA chain size in pages. nnn is a decimal number from 1 through 255. The size of a page is determined by:

- The value of mmm.
- The carriage control characters in the data that skip to channel 1.

mmm

Specifies the number of logical records in a page, when the data contains no carriage control characters. mmm is a decimal number from 1 through 255.

COMPACT = compaction-table-name
Specifies the compaction table for JES3 to use when sending a systems network architecture (SNA) data set to a SNA remote terminal. The compaction-table-name is a symbolic name defined by the installation during JES3 initialization. The name is 1 through 8 alphanumeric characters.

In the following cases, JES3 performs compaction using an installation default table, if defined, or sends the data without compacting it, if no table was defined. In all cases, JES3 writes a message to the console.

- No compaction table is specified.
- The specified compaction table is invalid.
- JES3 cannot find the specified compaction table.

If the remote printer does not support compaction, JES3 ignores the COMPACT parameter and sends the data without compacting it.

CONTROL = PROGRAM
CONTROL = SINGLE
CONTROL = DOUBLE
CONTROL = TRIPLE
Indicates either that the data records control printing or that the output is to be printed with single, double, or triple spacing.

PROGRAM
Indicates that each logical record in the sysout data set begins with a carriage control character. The carriage control characters can be in either the extended USASCII code or can be the actual channel command code. The carriage control characters are given in the *Data Administration Guide*.

SINGLE
Requests single spacing.

DOUBLE
Requests double spacing.

TRIPLE
Requests triple spacing.

COPIES = nnn
COPIES = (nnn,(group-value[,group-value]...))
COPIES = (group-value[,group-value]...)
Indicates how many copies of the sysout data set are to be printed. If a COPIES parameter is not specified, the default is 1.

nnn
Specifies how many copies of the sysout data set are to be printed; each copy will be in page sequence order. nnn is a number from 0 through 255. If you code COPIES = 0, JES3 does not print this data set. You can omit the parentheses if you code only nnn. JES3 ignores nnn if any group-values are specified.

group-value
Specifies how many copies of each page are to be printed before the next page is printed. Each group-value is a number from 1 through 255. You can code a maximum of eight group-values. Their sum must not exceed 255. The total copies of each page equals the sum of the group-values.

This subparameter is valid only for output on a 3800 Printing Subsystem. Group values override an nnn subparameter.

DEST = destination
Routes the output from the sysout data set to a printer. This parameter overrides the //*MAIN statement ORG parameter.

If you omit DEST, JES3 assigns the first available printer that is in the origin group and that fulfills all processing requirements. The origin group is the group of printers defined for the local or remote submitting location. If the job originated at a remote job processing (RJP) terminal, JES3 returns the output to the originating terminal group.

ANYLOCAL
Indicates any local printer that is being used for the output class specified in the SYSOUT parameter on the DD statement and that is attached to the global processor.

device-name
Requests a local device by a symbolic name defined by the installation during JES3 initialization. device-name is 1 through 8 alphanumeric or national ($, #, @) characters.

device-number
Specifies the 3-character device number.

group-name
Identifies a group of local devices, an individual remote station, or a group of remote stations by a symbolic name defined by the installation during JES3 initialization. group-name is 1 through 8 alphanumeric or national ($, #, @) characters.

nodename
Identifies a node by a symbolic name defined by the installation during JES3 initialization. nodename is 1 through 8 alphanumeric or national ($, #, @) characters.

remote
Identifies a remote work station or VM userid to which the receiving node directs output. remote is 1 through 8 characters.

(type)
Indicates a device classification. type is in the form **(gggssss)** where **ggg** is the general device classification and **ssss** is the specific device classification. The type must be enclosed in parentheses. The type must be defined by the installation during JES3 initialization. For example, type for a 3800 is (PRT3800).

EXTWTR = name
Identifies the external writer that is to process the sysout data set at the destination node. name is 1 through 8 alphanumeric characters and must identify a module defined to the remote JES3 node that is to execute the job. (Do not code NJERDR, it is reserved for JES3.)

FCB = image-name
FCB = 6
Specifies the forms control buffer (FCB) image JES3 is to use to guide printing of the sysout data set by a 1403 Printer, 3211 Printer, 3203 Printer Model 5, 4245 Printer, 4248 Printer, or 3800 Printing Subsystem, or by a printer supported by systems network architecture (SNA) remote job processing (RJP).

If the data set is to be produced on some other device, JES3 ignores the FCB parameter.

image-name
Identifies the FCB image. The name is 1 through 4 alphanumeric or national ($, #, @) characters and is the last characters of a SYS1.IMAGELIB member name:

- FCB2xxxx member for a 3211, 3203 model 5, or printer supported by SNA.
- FCB3xxxx member for a 3800.
- FCB4xxxx member for a 4248.

6
Indicates the standard FCB. JES3 uses the standard FCB specified at JES3 initialization.

Note: You cannot code both the CARRIAGE and FCB parameters on the same //*FORMAT PR statement.

FLASH = STANDARD
FLASH = overlay-name
FLASH = (overlay-name[,count])
Identifies the forms overlay to be used in printing the sysout data set on a 3800 Printing Subsystem and, optionally, to specify the number of copies on which the forms overlay is to be printed.

You can omit the parentheses if you code only an overlay-name. If you omit the count subparameter or specify a count of 0, JES3 flashes all copies with the specified overlay.

STANDARD
Indicates the standard forms flash overlay. JES3 uses the standard forms overlay specified at JES3 initialization.

overlay-name
Identifies the forms overlay frame that the operator is to insert into the printer before printing begins. The name is 1 through 4 alphanumeric or national ($, #, @) characters.

count
Specifies the number, 0 through 255, of copies that JES3 is to flash with the overlay, beginning with the first copy printed. Code a count of 0 to flash **all** copies.

Note: See the *Forms Design Reference Guide for the IBM 3800 Printing Subsystem* for information on designing and making forms overlays.

FORMS = STANDARD
FORMS = form-name
Indicates the forms on which the sysout data set is to be printed.

STANDARD
Indicates the standard form. JES3 uses the standard form specified at JES3 initialization.

form-name
Names the print forms. form-name is 1 through 8 alphanumeric characters.

MODIFY = module-name
MODIFY = (module-name[,trc])
Specifies a copy modification module that tells JES3 how to print the sysout data set on a 3800 Printing Subsystem. The module can specify how to replace blanks or data in the data set. You can omit the parentheses if you code only a module-name.

The module is defined and stored in SYS1.IMAGELIB using the IEBIMAGE utility program. See the *Data Administration: Utilities* for more information.

If you omit the trc subparameter, JES3 prints the data set with the first character-arrangement table coded in the CHARS parameter.

module-name
Identifies a copy modification module in SYS1.IMAGELIB. module-name is 1 through 4 alphanumeric or national ($, #, @) characters.

trc
Identifies which table-name in the CHARS parameter is to be used. This table reference character is 0 for the first table-name specified, 1 for the second, 2 for the third, or 3 for the fourth.

OVFL = ON
OVFL = OFF
Indicates whether or not the printer program should test for forms overflow.

Because the overflow test is a responsibility of the terminal package for the remote RJP terminal, JES3 ignores OVFL for remote job processing.

ON
Indicates that the printer program should eject whenever the end-of-forms indicator (channel 12) is sensed. ON is the default if the OVFL parameter is omitted.

OFF
Indicates that forms overflow control is not to be used.

PRTY = nnn
Specifies the priority at which the sysout data set enters the output queue. nnn is a decimal number from 0 through 255; 0 is the lowest priority while 255 is the highest.

STACKER=STANDARD
STACKER=S
STACKER=C
Requests a stacker for 3800 Printing Subsystem output.

STANDARD
Indicates the standard installation default. This default is specified at JES3 initialization.

S
Indicates the burster-trimmer-stacker, in which the output is burst into separate sheets.

C
Indicates the continuous forms stacker, in which the output is left in continuous fanfold.

THRESHLD=limit
Specifies the maximum size for the sysout data set. JES3 calculates the sysout data set size as the number of records multiplied by the number of copies requested. When this size exceeds the THRESHLD value, JES3 creates a new unit of work, on a data set boundary, and queues it for printing. Consequently, copies of the sysout data set may be printed simultaneously by different printers.

Use the THRESHLD parameter for jobs that generate many large sysout data sets. Grouping data sets as a single unit of work for an output service writer may decrease the time required for the output service writer to process the data sets.

The value specified in this parameter overrides the value specified during JES3 initialization.

limit
Specifies the maximum records for a single sysout data set. limit is a decimal number from 1 through 99999999. The default is 99999999.

TRAIN=STANDARD
TRAIN=train-name
Indicates the printer train to be used in printing the sysout data set. See Figure 10-2 on page 10-171 for the IBM-supplied trains. Because these trains are not standard machine features, verify that the installation has the required printer train before specifying it.

Do not code the TRAIN parameter for output destined for a remote job processing (RJP) terminal.

STANDARD
Indicates the standard installation default. This default is specified at JES3 initialization.

train-name
Specifies an installation-supplied printer train. Check with your installation for the names of trains.

Relationship to Sysout DD and OUTPUT JCL Statements

- JES3 ignores the processing options specified on a default //*FORMAT statement when a sysout DD statement explicitly or implicitly references an OUTPUT JCL statement.
- JES3 ignores the processing options specified on a default OUTPUT JCL statement when a //*FORMAT statement explicitly references a sysout DD statement.
- When a sysout DD statement explicitly references an OUTPUT JCL statement and a //*FORMAT statement explicitly references the same DD statement, the processing options from both the OUTPUT JCL and //*FORMAT statements apply. Two separate sets of output are created from the data set defined by the sysout DD statement; one according to the processing options on the OUTPUT JCL and DD statements, and the other according to the processing options on the //*FORMAT and DD statements.

Relationship to //*PROCESS Statement

JES3 accumulates //*FORMAT PR statements within a job and applies them to any JES3 //*PROCESS statement that is normally affected by a //*FORMAT PR statement.

Location in the JCL

Place all //*FORMAT PR statements for the job after the JOB statement and before the first EXEC statement.

Examples of the //*FORMAT PR Statement

```
//*FORMAT PR,DDNAME=STEP1.REPORT,COPIES=2
```

This statement requests two copies of the data set defined by sysout DD statement REPORT, which appears in STEP1 of this job. Any printer with standard forms, train, and carriage tape can be used.

```
//*FORMAT PR,DDNAME=,DEST=ANYLOCAL
```

This statement specifies that all sysout data sets not referenced by //*FORMAT PR statements are to be printed on any local printer.

//*FORMAT PU Statement

Purpose: Use the //*FORMAT PU statement to specify to JES3 processing instructions for sysout data sets that are punched. These instructions permit special processing of sysout data sets, such as:

- Multiple destinations.
- Multiple copies of output with different attributes.

You can code several //*FORMAT PU statements for one sysout data set to specify special requirements for different copies of the data set. You can also code a //*FORMAT PR statement for the same sysout data set, thereby both printing and punching it.

Note: The //*FORMAT PU statement applies only to sysout data sets punched by JES3. The statement is ignored for data sets sent to a TSO userid or processed by an external writer.

Syntax:

```
//*FORMAT PU,DDNAME={stepname.ddname                }[,parameter]...
                    {stepname.procstepname.ddname}

//*FORMAT PU,DDNAME=[,parameter]...

The parameters are:

 CHNSIZE={DS       }
         {(nnn[,mmm])}

 COMPACT=compaction-table-name

 COPIES=nnn

       {ANYLOCAL                }
       {device-name             }
       {device-number           }
       {group-name              }
 DEST= {nodename[.remote]       }
       {(type[,device-name ])   }
       {(type[,device-number])  }
       {(type[,group-name])     }

 EXTWTR=name

 FORMS={STANDARD  }
       {form-name}

 INT={YES}
     {NO }
```

The //*FORMAT PU statement consists of the characters //* in columns 1 through 3, FORMAT in columns 4 through 9, a blank in column 10, PU in columns 11 and 12, a comma in column 13, and parameters in columns 14 through 72. JES3 ignores columns 73 through 80.

Parameter Definition

PU

Indicates that this statement is associated with a sysout data set that is punched.

DDNAME=
DDNAME=stepname.ddname
DDNAME=stepname.procstepname.ddname

(null)

Specifies that the parameters on this //*FORMAT PU statement are the defaults for the job. These parameters then apply to all of the job's sysout data sets that are punched, except those covered by a //*FORMAT PU statement with DDNAME=ddname.

Nonspecific and Specific Statements: A //*FORMAT PU statement that contains DDNAME= is called **nonspecific**; a statement that contains DDNAME=ddname is called **specific**.

Overrides: Parameters coded on a nonspecific //*FORMAT PU statement are overridden by parameters coded on sysout DD statements or by parameters in the JES3 SYSOUT initialization statement.

stepname.ddname
stepname.procstepname.ddname

Identifies the DD statement that defines the sysout data set to be punched. Use form **stepname.ddname** to indicate DD statement, ddname, in step, stepname, in this job. Use form **stepname.procstepname.ddname** to indicate DD statement, ddname, in procedure step, procstepname, of a procedure that is called by a step, stepname, in this job. The ddname must match exactly the ddname on the DD statement. (See the example for the //*DATASET statement.) If the identified DD statement does not contain a SYSOUT parameter, JES3 ignores the //*FORMAT PU statement.

CHNSIZE=DS
CHNSIZE=(nnn[,mmm])

Gives the number of logical records to be transmitted to a work station as a systems network architecture (SNA) chain and indicates whether normal output checkpoints are to be taken for this sysout data set.

Note: This parameter is valid only when transmitting to a SNA work station.

Be careful in selecting subparameters, because each affects performance differently. Sending the data set as a SNA chain provides the best performance, but can cause duplicate data to be written to the output device if an operator intervention is required. The remote operator can eliminate duplicate data by issuing commands to reposition and restart the output writers.

When an end-of-chain indicator is sent in the data set, JES3 takes an output checkpoint. You can provide additional checkpoints for critical data by sending an end-of-chain indicator. For example, when punching bank checks, you can have an output checkpoint taken for each check by specifying each check as a SNA chain.

DS
Indicates that the sysout data set is to be sent as a single SNA chain and that JES3 is to take normal output checkpoints. DS is the default if the CHNSIZE parameter is omitted.

nnn
Specifies the SNA chain size in pages. nnn is a decimal number from 1 through 255. The size of a page is determined by the value you assign to mmm.

mmm
Specifies the number of logical records in a page. mmm is a decimal number from 1 through 255.

COMPACT = compaction-table-name
Specifies the compaction table for JES3 to use when sending a systems network architecture (SNA) data set to a SNA remote terminal. The compaction-table-name is a symbolic name defined by the installation during JES3 initialization. The name is 1 through 8 alphanumeric characters.

In the following cases, JES3 performs compaction using an installation default table, if defined, or sends the data without compacting it, if no table was defined. In all cases, JES3 writes a message to the console.

- No compaction table is specified.
- The specified compaction table is invalid.
- JES3 cannot find the specified compaction table.

If the remote punch does not support compaction, JES3 ignores the COMPACT parameter and sends the data without compacting it.

COPIES = nnn
Indicates how many copies of the sysout data set are to be punched. nnn is a number from 0 through 255. If you code COPIES = 0, JES3 does not punch this data set. If a COPIES parameter is not specified, the default is 1.

DEST = destination
Routes the output from the sysout data set to a punch. This parameter overrides the //*MAIN statement ORG parameter.

If you omit DEST, JES3 assigns the first available punch that is in the origin group and that fulfills all processing requirements. The origin group is the group of punches defined for the local or remote submitting location. If the job originated at a remote job processing (RJP) terminal, JES3 returns the output to the originating terminal group.

ANYLOCAL
Indicates any local punch that is being used for the output class specified in the SYSOUT parameter on the DD statement and that is attached to the global processor.

device-name
Requests a local device by a symbolic name defined by the installation during JES3 initialization. device-name is 1 through 8 alphanumeric or national ($, #, @) characters.

device-number
Specifies the 3-character device number.

group-name
Identifies a group of local devices, an individual remote station, or a group of remote stations by a symbolic name defined by the installation during JES3 initialization. group-name is 1 through 8 alphanumeric or national ($, #, @) characters.

nodename
Identifies node by a symbolic name defined by the installation during JES3 initialization. nodename is 1 through 8 alphanumeric or national ($, #, @) characters.

remote
Identifies a remote work station or VM userid to which the receiving node directs output. remote is 1 through 8 characters.

(type)
Indicates a device classification. type is in the form **(gggssss)** where **ggg** is the general device classification and **ssss** is the specific device classification. The type must be enclosed in parentheses. The type must be defined by the installation during JES3 initialization. For example, type for a 3525 is (PUN3525).

EXTWTR = name
Identifies the external writer that is to process the sysout data set at the destination node. name is 1 through 8 alphanumeric characters and must identify a module defined to the remote JES3 node that is to execute the job.

FORMS = STANDARD
FORMS = form-name
Indicates the forms on which the sysout data set is to be punched.

STANDARD
Indicates the standard form. JES3 uses the standard form specified at JES3 initialization.

form-name
Names the punch forms. form-name is 1 through 8 alphanumeric characters.

INT = YES
INT = NO
Specifies whether or not the output is to be interpreted. If the INT parameter is omitted, the default is NO.

YES
Requests that JES3 try to punch the sysout data set on a 3525 Card Punch (PUN3525I) with a Multiline Card Print feature.

Note: If the DEST parameter does not send output to a 3525I, JES3 ignores INT = YES, if specified.

NO
Requests that the cards not be interpreted.

JES3: //*FORMAT PU

Relationship to Sysout DD and OUTPUT JCL Statements

- JES3 ignores the processing options specified on a default //*FORMAT statement when a sysout DD statement explicitly or implicitly references an OUTPUT JCL statement.
- JES3 ignores the processing options specified on a default OUTPUT JCL statement when a //*FORMAT statement explicitly references a sysout DD statement.
- When a sysout DD statement explicitly references an OUTPUT JCL statement and a //*FORMAT statement explicitly references the same DD statement, the processing options from both the OUTPUT JCL and //*FORMAT statements apply. Two separate sets of output are created from the data set defined by the sysout DD statement; one according to the processing options on the OUTPUT JCL and DD statements, and the other according to the processing options on the //*FORMAT and DD statements.

Relationship to //*PROCESS Statement

JES3 accumulates //*FORMAT PU statements within a job and applies them to any JES3 //*PROCESS statement that is normally affected by a //*FORMAT PU statement.

Location in the JCL

Place all //*FORMAT PU statements for the job after the JOB statement and before the first EXEC statement.

Example of the //*FORMAT PU Statement

```
//*FORMAT PU,DDNAME=STEP2.PUNCHOUT,DEST=PU1,FORMS=RED-STRP
```

This statement requests that one copy of the data set defined by sysout DD statement PUNCHOUT in STEP2 of this job be punched on device PU1. Before processing, the operator is requested to insert **RED-STRP** cards into the punch.

//*MAIN Statement

Purpose: Use the //*MAIN statement to define the processor requirements for the current job. Many of the parameters are used to override parameters on the JES3 STANDARDS initialization statement.

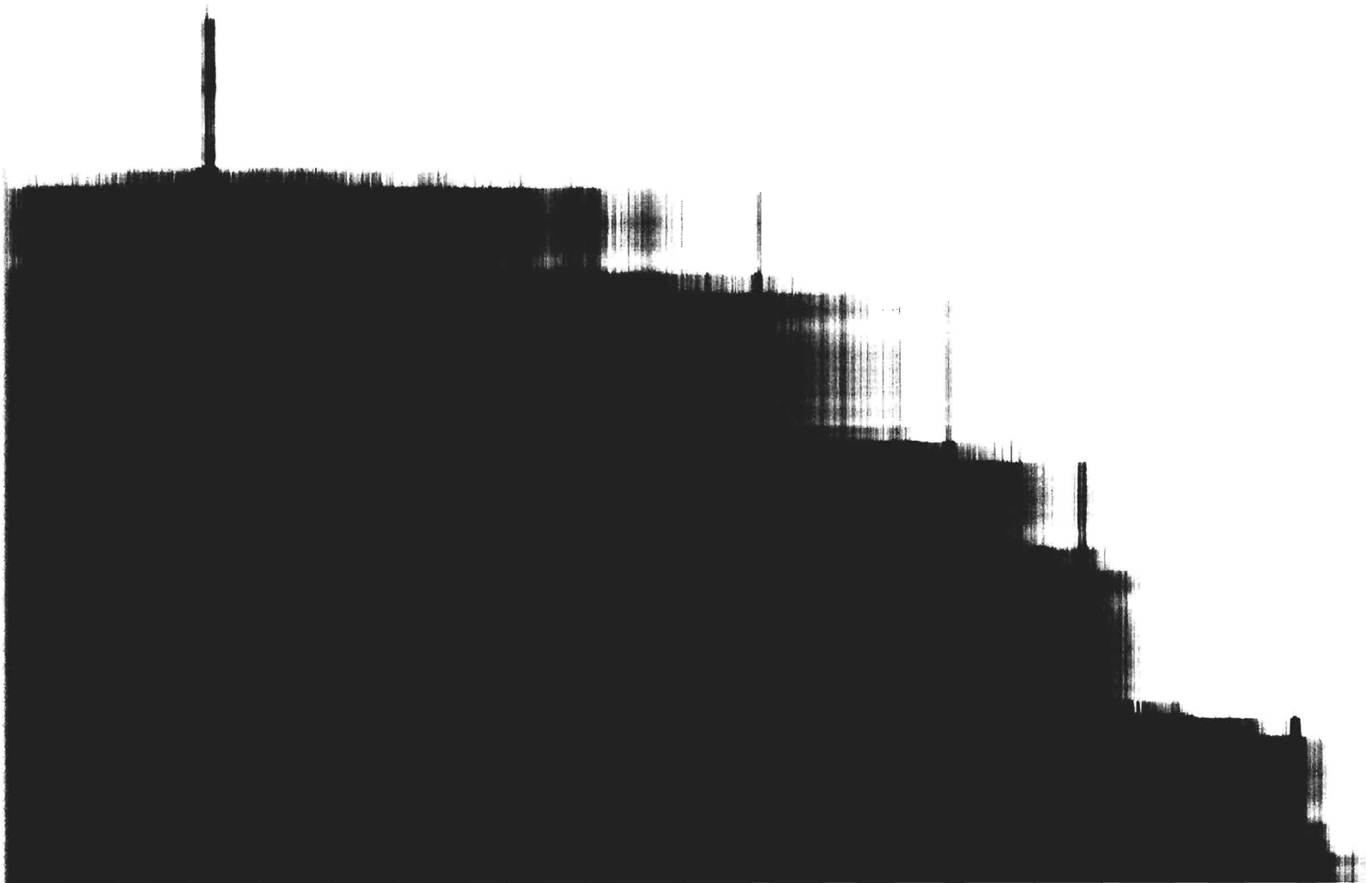
Note: If any parameter is misspelled or contains an invalid value, JES3 writes the following to the JESMSG data set: the //*MAIN statement, the relative error position on the statement, and an error message. Then JES3 abnormally terminates the job.

Syntax:

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

The parameters are:

ACMAIN=processor-id

BYTES=  ([nnnnnn][,WARNING][,mmm])
        ([nnnnnn][,W][,mmm])
        ([nnnnnn][,CANCEL])
        ([nnnnnn][,C])
        ([nnnnnn][,DUMP])
        ([nnnnnn][,D])

CARDS=  ([nnnn][,WARNING][,mmm])
        ([nnnn][,W][,mmm])
        ([nnnn][,CANCEL])
        ([nnnn][,C])
        ([nnnn][,DUMP])
        ([nnnn][,D])

CLASS=class-name

DEADLINE= (time,type[,date])
          (time,type[,rel,cycle])

EXPDTCHK= YES
          NO

FAILURE=  RESTART
          CANCEL
          HOLD
          PRINT

FETCH=  ALL
        NONE
        SETUP
        (ddname[,ddname]...)
        /(ddname[,ddname]...)

HOLD=   YES
        NO

IORATE=  MED
         HIGH
         LOW

JOURNAL=  YES
          NO
```

JES3: //*MAIN

```
LINES=  ([nnnn][,WARNING][,mmm])
        ([nnnn][,W][,mmm])
        ([nnnn][,CANCEL])
        ([nnnn][,C])
        ([nnnn][,DUMP])
        ([nnnn][,D])

LREGION=nnnnK

MSS=  {JOB}
      {HWS}

ORG=  {group-name       }
      {nodename[.remote]}

PAGES=  ([nnnnnnnn][,WARNING][,mmm])
        ([nnnnnnnn][,W][,mmm])
        ([nnnnnnnn][,CANCEL])
        ([nnnnnnnn][,C])
        ([nnnnnnnn][,DUMP])
        ([nnnnnnnn][,D])

PROC=  {ST}
       {xx}

RINGCHK=  {YES}
          {NO }

SETUP=  JOB
        HWS
        THWS
        DHWS
        (stepname.ddname[,stepname.ddname]...)
        (stepname.procstepname.ddname[,stepname.procstepname.ddname]...)
        /(stepname.ddname[,stepname.ddname]...)
        /(stepname.procstepname.ddname[,stepname.procstepname.ddname]...)

SPART=partition-name

SYSTEM=  ANY
         JGLOBAL
         JLOCAL
         (main-name[,main-name]...)
         /(main-name[,main-name]...)

THWSSEP=  IGNORE
          PREFER
          REQUIRE

TRKGRPS=(primary-qty,second-qty)

TYPE=  {ANY}
       {VS2}

UPDATE=(dsname[,dsname]...)

USER=userid
```



JES3: //*MAIN

The //*MAIN statement consists of the characters //* in columns 1 through 3, MAIN in columns 4 through 7, a blank in column 8, and parameters in columns 9 through 72. JES3 ignores columns 73 through 80.

Parameter Definition

ACMAIN = processor-id

Identifies the job with the specified processor, even though the job was not submitted from or executed on that processor. This parameter has no effect if it specifies the processor that the job runs on. ACMAIN allows:

- Sysout data sets to be sent to a userid attached to the specified processor. The userid must be named in the USER parameter. The ACMAIN parameter applies to all sysout data sets for the job.
- A job termination message to be sent automatically to a userid attached to the specified processor. The userid must be named in the JOB statement NOTIFY parameter.

processor-id

Requests a processor in the complex.

BYTES = ([nnnnnn][,WARNING][,mmm])
BYTES = ([nnnnnn][,W][,mmm])
BYTES = ([nnnnnn][,CANCEL])
BYTES = ([nnnnnn][,C])
BYTES = ([nnnnnn][.DUMP])
BYTES = ([nnnnnn][,D])

Specifies the maximum number of bytes of data to be spooled from this job's sysout data sets and the action to be taken if the maximum is exceeded.

If BYTES is not specified, the installation default for this job class applies.

nnnnnn

Specifies the number of bytes in thousands. nnnnnn is 1 through 6 decimal numbers from 1 through 999999.

WARNING or W

operator when 100,000 bytes of sysout data is reached. Subsequent warning messages are sent when each additional 20 percent of 100,000 is reached (at 120,000 bytes, 140,000 bytes, and so on). Messages are sent until the job ends or the operator cancels the job.

CANCEL or C
If the maximum is exceeded, requests that JES3 cancel the job.

DUMP or D
If the maximum is exceeded, requests that JES3 cancel the job and ask for a storage dump.

CARDS=([nnnn][,WARNING][,mmm])
CARDS=([nnnn][,W][,mmm])
CARDS=([nnnn][,CANCEL])
CARDS=([nnnn][,C])
CARDS=([nnnn][,DUMP])
CARDS=([nnnn][,D])
Specifies the maximum number of cards to be punched from this job's sysout data sets and the action to be taken if the maximum is exceeded.

If you specify CARDS=0 the zero applies only to the quantity of punched output; it does **not** cancel the action to be taken if the maximum is exceeded. If a record is then sent to a punch, JES3 will warn, cancel, or dump, depending on the second parameter.

Note: When punching dump output, JES3 ignores CARDS=0.

If CARDS is not specified, the installation default for this job class is used.

nnnn
Specifies the number of cards in hundreds. nnnn is 1 through 4 decimal numbers from 1 through 9999.

WARNING or W
If the maximum is exceeded, requests that JES3 issue an operator warning message and continue processing.

Any subsequent messages about this parameter will reflect the number specified on the STANDARD initialization statement or the system default, **not** the maximum specified in the CARDS parameter.

CANCEL or C
If the maximum is exceeded, requests that JES3 cancel the job.

DUMP or D
If the maximum is exceeded, requests that JES3 cancel the job and ask for a storage dump.

CLASS = class-name
Specifies the job class for this job. class-name is 1 through 8 characters.

If the desired class-name is a single-character, you can specify it on the //*MAIN statement or the JOB statement.

JES3 uses the following, in override order, to assign the job to a class:

1. //*MAIN statement CLASS parameter
2. JOB statement CLASS parameter
3. The default class, which is defined during JES3 initialization.

If neither CLASS nor LREGION is specified, JES3 determines the logical region size based on initialization parameters.

DEADLINE = (time,type[,date])
DEADLINE = (time,type[,rel,cycle])
Specifies when the job is required.

When you specify the current date but submit the job after the specified time, JES3 changes the priorities to make the job the same priority level it would have if it had been submitted before the deadline but not completed.

Warning: Deadline scheduling can interfere with dumping a portion of the job queue. For example, if JOB A is waiting to be scheduled, has a priority of 7, and, in one minute, is due to have its priority increased to 9, JOB A could be missed by dump job processing, if the dump job facility is dumping the entire job queue and currently dumping priority 8 jobs. The dump job facility processes the jobs with the highest priority first. If the dump job facility does not finish processing priority 8 jobs before JOB A becomes priority 9, JOB A will not be dumped.

Deadline scheduling information is not sent with a job when the job is transferred via NJE to another node; the destination node may use different deadline scheduling algorithms, if any.

time
Specifies the deadline time, expressed as one of the following:

nM
The job is to be scheduled within n minutes. n is 1 through 4 numbers from 0 through 1440.

nH
The job is to be scheduled within n hours. n is 1 or 2 numbers from 0 through 24.

hhhh
The job is to be scheduled by the time of day, hhhh, in 24-hour clock time (0800 is 8:00 a.m.). hhhh is from 0000 (start of the day) through 2400 (end of the day).

type
Identifies the deadline algorithm. The deadline algorithm is defined by the installation, controls how the job's priority is increased, and is one character: A through Z or 0 through 9. If the specified algorithm is not defined, JES3 abnormally terminates the job.

date
Specifies the date when the time parameter takes effect. The date is in the format **mmddyy**, where mm is the month (01-12), dd the day (01-31), and yy the year (00-99).

If both date and rel,cycle are omitted, JES3 assumes (1) the current date, if the deadline time is later in the day, or (2) the next day's date, if the deadline time has already past today.

rel
Specifies on which day within a cycle the deadline falls. rel is 1 through 3 numbers from 1 through 366. The value of rel depends on the specified cycle, as follows:

- WEEKLY: Sunday is day 1; Saturday is day 7. If rel is greater than 7, it defaults to 7.
- MONTHLY: Day 1 is the first day of the month. Days 29, 30, and 31 are treated as the last day of the month. If rel is greater than 31, it defaults to 31.
- YEARLY: Day 1 in January 1; day 365 is December 31, for regular years, and day 366 is December 31, for leap years. If rel is greater than 365, it defaults to 365 for regular years or 366 for leap years.

cycle
Specifies the length of a cycle. cycle is coded as WEEKLY, MONTHLY, or YEARLY.

For example, DEADLINE=(1200,B,1,WEEKLY) indicates that the job reaches its deadline at 12 noon on Sunday.

EXPDTCHK=YES
EXPDTCHK=NO
Indicates whether or not JES3 is to perform expiration date checking for scratch output tape volumes with IBM standard labels (SL).

YES
Requests expiration date checking. Tape volumes premounted for SL scratch requests must have expired dates.

NO
Requests that expiration dates not be checked.

FAILURE=RESTART
FAILURE=CANCEL
FAILURE=HOLD
FAILURE=PRINT

Indicates the job recovery option to be used if the system fails. If you do not code a FAILURE parameter on the //*MAIN statement, JES3 assigns the job the default failure option, which is defined during JES3 initialization for each job class. (See also the RD parameter on the JOB statement.)

RESTART

Requests that JES3 restart the job when the failing processor is restarted. Do not specify RESTART for jobs that use the DEQ at DEMOUNT facility for tape volumes.

CANCEL

Requests that JES3 print the job and then cancel the job.

HOLD

Requests that JES3 hold the job for restart.

PRINT

Requests that JES3 print the job and then hold the job for restart.

FETCH=ALL
FETCH=NONE
FETCH=SETUP
FETCH=(ddname[,ddname]...)
FETCH=/(ddname[,ddname]...)

Determines the fetch messages that will be issued to the operator for disk and tape volumes for this job.

If FETCH is not specified, the installation default for this job class applies.

ALL

Requests that JES3 issue fetch messages to the operator for all removable volumes specified in DD statements that request JES3-setup devices. This subparameter does not apply to permanently resident volumes.

NONE

Requests that JES3 not issue fetch messages.

SETUP

Requests that JES3 issue fetch messages to the operator for the volumes specified in all DD statements identified in the //*MAIN SETUP parameter. If you code FETCH=SETUP without also coding the //*MAIN SETUP parameter, JES3 will issue fetch message as though you had specified FETCH=ALL.

ddname

Requests that JES3 issue fetch messages for only the volumes specified in DD statement ddname.

If you code a list of ddnames and the list cannot be contained on a single statement, FETCH= must be repeated on the continuation statement.

/ddname
Requests that JES3 **not** issue fetch messages for any volumes specified in DD statement ddname.

HOLD=YES
HOLD=NO

YES
Indicates that the job is to enter the system in operator-hold status and be withheld from processing until the operator requests its release. However, if an error occurs during input service processing, the job is not held for operator intervention.

This parameter has the same function as TYPRUN=HOLD on the JOB statement.

NO
Indicates that the job is to enter the system normally. Processing does not require operator intervention. If the HOLD parameter is omitted, NO is the default.

IORATE=MED
IORATE=HIGH
IORATE=LOW
Indicates the I/O-to-processor ratio for a job. Use this parameter to balance the mixture of jobs selected for execution on the processor.

If you do not code an IORATE parameter on the //*MAIN statement, JES3 assigns the job the default I/O-to-processor ratio, which is defined during JES3 initialization for each job class.

JOURNAL=YES
JOURNAL=NO
Indicates whether or not JES3 is to create a job journal for the job.

If JOURNAL is omitted, JES3 uses an installation default specified at initialization.

YES
Indicates that the job is to have a job journal.

NO
Indicates that the job is not to have a job journal.

LINES=([nnnn][,WARNING][,mmm])
LINES=([nnnn][,W][,mmm])
LINES=([nnnn][,CANCEL)
LINES=([nnnn][,C])
LINES=([nnnn][,DUMP])
LINES=([nnnn][,D])
Indicates the maximum number of lines of data to be printed from this job's sysout data sets and the action to be taken if the maximum is exceeded.

If you specify LINES=0 the zero applies only to the number of lines; it does **not** cancel the action to be taken if the maximum is exceeded. If a record is sent to be printed, JES3 will warn, cancel, or dump, depending on the second parameter.

Note: JES3 ignores any line count specification when printing the output for a SYSABEND or SYSUDUMP sysout data set.

If LINES is not specified, the installation default for this job class applies.

nnnn
Specifies the number of lines, in thousands. nnnn is 1 through 4 decimal numbers from 1 through 9999.

WARNING or W
If the maximum is exceeded, requests that JES3 issue an operator warning and continue processing.

Any messages about this parameter following the warning message will reflect the number specified on the STANDARD initialization statement or the system default, **not** the maximum specified in the LINES parameter.

mmm
Specifies the frequency that an operator warning message is to be issued after the maximum specified by nnnn is exceeded. mmm is a multiple of 10 in the range 10 to 100. mmm is a percentage of nnnn that is used to calculate the number of additional lines between warning messages. For example, if LINES=(100,W,20) is specified, the first warning message is sent to the operator when 100,000 lines of sysout data is reached. Subsequent warning messages are sent when each additional 20 percent of 100,000 is reached (at 120,000 lines, 140,000 lines, and so on). Messages are sent until the job ends or the operator cancels the job.

CANCEL or C
If the maximum is exceeded, requests that JES3 cancel the job.

DUMP or D
If the maximum is exceeded, requests that JES3 cancel the job and ask for a storage dump.

LREGION=nnnnK
Specifies the approximate size of the largest step's working set in real storage during execution. LREGION (logical region) is used by JES3 to improve scheduling on the processor. The nnnn is 1 through 4 decimal numbers that indicate the size in kilobytes (1 kilobyte = 1024 bytes).

If neither CLASS nor LREGION is coded, JES3 determines the logical region size based on initialization parameters.

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

MSS=JOB
MSS=HWS
Indicates that the job uses mass storage system (MSS) and specifies how JES3 should allocate the MSS virtual units. This parameter overrides the installation default defined at JES3 initialization.

JOB
Specifies that each request for an MSS volume is to be assigned to a separate virtual unit.

HWS
Specifies that virtual units are to be reused in subsequent job steps in order to minimize the number of units needed for the job.

Note: The MSS and SETUP parameters on the //*MAIN statement can specify different types of allocation. For example, MSS can specify JOB while SETUP can specify HWS.

ORG=group-name
ORG=nodename[.remote]
Indicates that the job's sysout data sets are to be directed to the named group or network node. Otherwise, the job's sysout data sets are directed to the group of devices or node from which the job originated.

group-name
Specifies an origin group.

nodename
Specifies a network node. nodename is 1 through 8 characters.

remote
Specifies a remote work station or VM userid. remote is 1 through 8 characters and must be separated from the nodename by a period.

Overriding an ORG Parameter: If you do not want a particular data set in the job to go to the destination on the ORG parameter, change its destination in one of the following ways:

- If the sysout data set is not scheduled to a **held** class, you can override the ORG parameter destination with the DEST parameter on a //*FORMAT, OUTPUT JCL, or DD statement.
- If the sysout data set is scheduled to a **held** class, you can override the ORG parameter destination with the DEST parameter on an OUTPUT JCL, or DD statement.

PAGES=([nnnnnnnn][,WARNING][,mmm])
PAGES=([nnnnnnnn][,W][,mmm])
PAGES=([nnnnnnnn][,CANCEL])
PAGES=([nnnnnnnn][,C])
PAGES=([nnnnnnnn][,DUMP])
PAGES=([nnnnnnnn][,D])
Indicates the maximum number of pages to be printed for this job's sysout data sets and the action to be taken if the maximum is exceeded.

If PAGES is not specified, the installation default for this job class applies.

nnnnnnnn
Specifies the number of pages. nnnnnnnn is 1 through 8 decimal numbers from 1 through 16777215.

WARNING or W
If the maximum is exceeded, requests that JES3 issue an operator warning message and continue processing.

Any messages about this parameter following the warning message will reflect the number specified on the STANDARD initialization statement or the system default value, **not** the maximum specified in the PAGES parameter.

mmm
Specifies the frequency that an operator warning message is to be issued after the maximum specified by nnnnnnnn is exceeded. mmm is a multiple of 10 in the range 10 to 100. mmm is a percentage of nnnnnnnn that is used to calculate the number of additional pages between warning messages. For example, if PAGES=(1000,W,20) is specified, the first warning message is sent to the operator when 1,000 pages of sysout data is reached. Subsequent warning messages are sent when each additional 20 percent of 1,000 is reached (at 1,200 pages, 1,400 pages, and so on). Messages are sent until the job ends or the operator cancels the job.

CANCEL or C
If the maximum is exceeded, requests that JES3 cancel the job.

DUMP or D
If the maximum is exceeded, requests that JES3 cancel the job and ask for a storage dump.

PROC=ST
PROC=xx
Names the procedure library that the system is to search for cataloged procedures called by EXEC statements in the job. If a procedure cannot be found in the named library, JES3 abnormally terminates the job.

If this parameter is omitted, the default depends on the source of the job. If the job is submitted as a batch job, the default is ST. If the job is submitted from an internal reader, the default may be another procedure library, as specified by the installation.

ST
Indicates the standard procedure library: SYS1.PROCLIB.

xx
Identifies the last 2 characters of the ddname of a procedure library. xx is defined by the installation. If this parameter is coded, only the specified library is searched; SYS1.PROCLIB is not searched.

RINGCHK=YES
RINGCHK=NO
Indicates whether or not JES3 is to check the status of the tape reel ring for tape devices set up by JES3.

YES
Indicates that a validation check is to be made. If the RINGCHK parameter is omitted, YES is the default.

NO
Indicates that ring checking is to be by-passed for this job.

SETUP=JOB
SETUP=HWS
SETUP=THWS
SETUP=DHWS
SETUP=(stepname.ddname[,stepname.ddname]...)
SETUP=(stepname.procstepname.ddname[,stepname.procstepname.ddname]...)
SETUP=/(stepanme.ddname[,stepname.ddname]...)
SETUP=/(stepanme.procstepname.ddname[,stepname.procstepname.ddname]...)
Modifies the standard setup algorithm used in assigning devices to a job before its execution.

If SETUP is omitted, JES3 assigns mountable tape and disk volumes based on an installation default defined at initialization.

JOB
Requests job setup, which is allocation of all JES3-managed devices required in the job before the job executes. JES3 mounts the initial volumes necessary to run all steps before the job executes. JOB overrides the SETUP parameter on the JES3 STANDARDS initialization statement.

HWS
Requests high watermark setup, which is allocation of the minimum number of devices required to run the job. The minimum number is equal to the greatest number of devices of each type needed for any one job step. High watermark setup does not cause premounting of all mountable volumes.

THWS
Requests high watermark setup for tapes but job setup for disks.

DHWS
Requests high watermark setup for disks but job setup for tapes.

stepname.ddname
stepname.procstepname.ddname
Specifies explicit setup, which is allocation of the volumes needed for a DD statement before the job executes. JES3 premounts the indicated volumes. When requesting explicit setup, specify enough devices so that JES3 can allocate all the required devices at any one time. If too few devices are specified, JES3 cancels the job.

Use form **stepname.ddname** to indicate DD statement, ddname, in step, stepname, in this job. Use form **stepname.procstepname.ddname** to indicate DD statement, ddname, in procedure step, procstepname, of a procedure that is called by a step, stepname, in this job. The ddname must match exactly the ddname on the DD statement. (See the example for the //*DATASET statement.)

If you code a list of ddnames and the list cannot be contained on a single statement, SETUP= must be repeated on the continuation statement.

/stepname.ddname
/stepname.procstepname.ddname
Requests that JES3 **not** explicitly set up any volumes specified in DD statement ddname.

SPART = partition-name
Indicates the spool partition in which JES3 is to allocate spool space to this job.

partition-name
Specifies the name of the spool partition. **partition-name** is 1 through 8 characters and must match a partition name specified during JES3 initialization. If the name does not match, JES3 ignores the SPART parameter and uses the installation default.

The SPART parameter does not affect allocation for the sysout data sets for the job; these data sets always go to the spool partitions specified during JES3 initialization for the output classes.

If SPART is not specified, JES3 allocates spool data sets to a partition, as follows, in override order:

1. The spool partition for the job's class.
2. The spool partition for the processor executing the job.
3. The default spool partition.

SYSTEM = ANY
SYSTEM = JGLOBAL
SYSTEM = JLOCAL
SYSTEM = (main-name[,main-name]...)
SYSTEM = /(main-name[,main-name]...)
Indicates the processor that is to execute this job. If a specific processor is named, the processor name must also be specified on the CLASS initialization statement for the job class.

ANY
Indicates any global or local system that satisfies the job's requirements.

JGLOBAL
Indicates that the job is to run on the global processor only.

JLOCAL
Indicates that the job is to run on a local processor only.

main-name
Indicates that the job is to run on the named processor or processors.

/main-name
Indicates that the job is not to run on the named processor or processors.

Need for SYSTEM Parameter: If you omit a SYSTEM parameter, the job runs on the processor used for the job's class. Usually a SYSTEM parameter is not needed. However, if any DD statement UNIT parameter in the job specifies a device-number, a SYSTEM parameter must be coded.

Parameter Agreements: The following parameters must be consistent with the SYSTEM parameter or JES3 will terminate the job:

- CLASS parameter on the JOB or //*MAIN statement. The requested processor must be assigned to execute jobs in the specified class.
- All devices specified on DD statement UNIT parameters must be available to the requested processor.
- TYPE parameter on the //*MAIN statement must specify the system running on the requested processor.
- Dynamic support programs requested on //*PROCESS statements must be able to be executed on the requested processor.

THWSSEP=IGNORE
THWSSEP=PREFER
THWSSEP=REQUIRE

Indicates whether or not you want scratch tape requests and specific tape requests separated during high watermark processing. This parameter is valid only if high watermark setup (HWS or THWS) is specified on the SETUP parameter or defined at JES3 initialization.

Use this parameter to direct scratch and specific tape requests to different tape drives (for example, you may want JES3 to allocate only scratch tape requests to an IBM 3480 that is equipped with an automatic cartridge loader).

If you omit THWSSEP, JES3 uses an installation default defined at initialization.

IGNORE
Specifies that JES3 is not to separate scratch and specific tape requests during high watermark processing. Both scratch and specific tape requests can be allocated on the same tape drive.

PREFER
Specifies that JES3 attempt to allocate scratch and specific tape requests on separate tape drives without allocating additional devices. If JES3 cannot separate the requests, scratch and specific tape requests are allocated on the same tape drive.

REQUIRE
Specifies that JES3 should not allocate scratch and specific tape requests on the same tape drive, even if JES3 must allocate additional tape drives to satisfy the request.

TRKGRPS=(primary-qty,second-qty)

Specifies the number of track groups to be assigned to the job. A track group is a number of spool space allocation units. The size of the track group is defined in the GRPSZ parameter on the JES3 BUFFER or SPART initialization statement.

primary-qty
Specifies the number of track groups to be initially allocated. This quantity is one decimal number from 1 through 9.

second-qty
Specifies the number of track groups to be allocated when the currently allocated groups are filled and more space is needed. This quantity is one decimal number from 1 through 9.

The //*MAIN TRKGRPS parameter overrides a TRKGRPS parameter on the CLASS or MAINPROC initialization statement. However, when a sysout DD statement specifies an output class, the TRKGRPS parameter for that output class overrides the //*MAIN TRKGRPS parameter.

TYPE = ANY
TYPE = VS2
Indicates the control program that is to execute this job. If you omit a TYPE parameter, the job runs under the control program used for the job's class.

ANY
Indicates that JES3 is to use any control program that satisfies the job's requirements. In present systems, JES3 schedules the job on MVS.

VS2
Indicates that JES3 is to schedule the job on MVS.

UPDATE = (dsname[,dsname]...)
Identifies the procedure library data set(s) that this job is to update. This parameter causes all jobs using the identified data set and any concatenated data sets to be held until the update is complete. See *JES3 Initialization and Tuning* for information about updating procedure libraries.

dsname
Specifies the data set name. The identified data set cannot be concatenated to another data set.

USER = userid
Identifies the job with the specified TSO user, even though the job was not submitted via TSO by that user. USER allows:

- The TSO userid, interacting with a global or local processor, to issue the TSO OUTPUT command to access sysout data sets from the job. 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.

- The TSO userid, interacting with any processor, to inquire about the status of the job or to cancel the job.

userid
Identifies a TSO user. userid is 1 through 7 alphanumeric or national ($, #, @) characters.

JES3: //*MAIN

Location in the JCL

Place the //*MAIN statement for a job after the JOB statement and before the first EXEC statement.

When you specify **ORG** on a //*MAIN statement, place the //*MAIN statement before all //*FORMAT statements that do not contain a DEST parameter. If JES3 does not process the ORG parameter before the //*FORMAT statements, JES3 uses the default destination for the //*FORMAT statements; their output is sent to the node where the job entered the system.

When you specify **ORG** on a //*MAIN statement that is part of a remote job, place the //*MAIN statement immediately after the second JOB statement.

Examples of the //*MAIN Statement

```
//*MAIN SYSTEM=SY1,LINES=(5,C),SETUP=HWS,
//*FAILURE=RESTART,DEADLINE=(0800,A,3,WEEKLY)
```

The job executes on processor SY1. It is estimated to produce not more than 5000 lines of printed output; if the output exceeds 5000 lines, JES3 is to cancel the job. HWS specifies high watermark setup, so JES3 is to allocate the minimum number of devices required for this job. If the system fails, JES3 is to restart the job on the processor SY1. JES3 is to complete this job by 8 a.m. on Tuesday (Tuesday is day number 3) by adjusting the job's scheduling priority using the installation-defined A-type deadline scheduling parameters.

```
//*MAIN ACMAIN=2,USER=GARYHIL
```

If this statement appears in a job entered from any TSO userid on any processor in the complex, then the job's sysout data sets would go to TSO userid GARYHIL on processor 2.

//*NET Statement

Purpose: Use the //*NET statement to define the dependencies between jobs in a dependent job control (DJC) network. JES3 sets up a network of dependent jobs and executes them in a specific order. (Once set up, the structure of a DJC network cannot be changed unless all of the jobs in the network are resubmitted.)

Syntax:

```
//*NET  {NETID}=name[,parameter]...
        {ID   }

The parameters are:

 {ABCMP}={NOKP}
 {AC   } {KEEP}

 {{ABNORMAL}={D}}
 {{AB      } {F}}
 {{        } {R}}
 {{NORMAL  }={D}}
 {{NC      } {F}}
 {{        } {R}}

 DEVPOOL=({ANY}[,device-name,n]...[,SDGxx]...)
          {NET}

 DEVRELSE={YES}
          {NO }

 {NETREL}=(netid,jobname)
 {NR    }

 {NHOLD}=n
 {HC   }

 {NRCMP}={HOLD}
 {PC   } {NOHO}
         {FLSH}

 {OPHOLD}={NO }
 {OH    } {YES}

 {RELEASE}=(jobname[,jobname]...)
 {RL     }

 {RELSCHCT}=n
 {RS      }
```

The //*NET statement consists of the characters //* in columns 1 through 3, NET in columns 4 through 6, a blank in column 7, and parameters in columns 8 through 72. JES3 ignores columns 73 through 80.

Parameter Definition

NETID = name
Specifies the name of the DJC network for this job. name is 1 through 8 characters; the first character must be alphabetic.

All jobs put into the system with the same NETID name form a DJC network. To add a job to an existing DJC network, specify the NETID name for that job.

ABCMP = NOKP
ABCMP = KEEP
Indicates what action JES3 is to take if the job abnormally terminates.

NOKP
Indicates that JES3 is to purge the DJC network if the job abnormally terminates and has not been resubmitted by the time the other jobs in the network have completed. JES3 purges the network unless successor jobs or subnetworks are missing. If the ABCMP parameter is omitted, NOKP is the default.

KEEP
Indicates that the DJC network is to be kept in the system until (1) the job is resubmitted and completes normally or (2) the operator forces the network from the system. Use KEEP to make sure that the network is not purged until the operator takes proper action.

Note: If the job abnormally terminates, you can resubmit it to the DJC network, and the network will be retained until the job completes.

ABNORMAL = D
ABNORMAL = F
ABNORMAL = R
NORMAL = D
NORMAL = F
NORMAL = R
Indicates the action JES3 is to take for this job when any predecessor job completes execution normally or abnormally. If the ABNORMAL parameter is omitted, the default is R, and, if the NORMAL parameter is omitted, the default is D.

D
Requests that JES3 decrease this job's NHOLD count, which indicates the number of predecessors for this job. When the NHOLD count becomes zero, JES3 can schedule this job.

F
Requests that JES3 flush this job and its successor jobs from the system. JES3 cancels the job, prints any output, and cancels all successor jobs presently in the system, regardless of their normal or abnormal specifications. However, JES3 admits into the system all successor jobs that enter after the DJC network has been flushed. To flush those jobs, the operator must cancel the jobs or the network.

R
Requests that JES3 retain this job in the system and not decrease the NHOLD count. R suspends the job and its successor jobs from scheduling until either the predecessor job is resubmitted or the operator decreases the NHOLD count.

DEVPOOL = (ANY[,device-name,n]...[,SDGxx]...)
DEVPOOL = (NET[,device-name,n]...[,SDGxx]...)

Identifies devices to be dedicated to this DJC network. The system allocates these devices only to jobs in the network. The DEVPOOL parameter should be coded on the //*NET statement that establishes the network; it is ignored on other //*NET statements.

ANY

Indicates that jobs in the network can use any dedicated or undedicated device. JES3 tries to allocate from the dedicated pool before allocating any undedicated devices.

NET

Indicates that jobs can use only devices dedicated to the network.

device-name,n

Identifies a dedicated device. Code as many device-names with numbers as will fit on one statement. **device-name** specifies (1) a device name defined to JES3 by the installation during initialization or (2) a device-type specified in the UNIT parameter of an IODEVICE system generation macro instruction. See *Installation: System Generation* for a list of IBM device types. **n** is the number of named devices. n is a number from 1 through 32767.

SDGxx

Identifies a mass storage system (MSS) staging drive group to be pooled (fenced) for the DJC network. xx is the staging drive group number.

DEVRELSE = YES
DEVRELSE = NO

Indicates when devices dedicated to the DJC network are to be released. The DEVRELSE parameter can be coded in several jobs in the network, but must not be coded in the first job. If no network job containing DEVRELSE = YES completes, the system releases the devices when it purges the network.

YES

Requests that JES3 release all devices at the end of this job. Completion of any job that specified DEVRELSE = YES causes the devices dedicated to the network to be released.

NO

Requests that JES3 release all devices only when the last job in the network ends.

NETREL = (netid,jobname)

Indicates that this job must be executed before the named job in another DJC network can be executed. The NETREL parameter can be specified only once for each job of a DJC network.

netid

Identifies the NETID for the successor job.

jobname

Names the JOB statement for the successor job.

NHOLD=n
Indicates the number of predecessor job completions required before this job can be released for scheduling. The predecessor number can include jobs from another DJC network. **n** is a number from 0 through 32767.

When the predecessor number reaches 0, the job is scheduled for execution. The system reduces this number:

- When each predecessor job completes execution.
- By operator command.
- When a program in a predecessor job issues an assembler DJC WTO macro. See *Supervisor Services and Macro Instructions* for details.

If you specify NHOLD=0 or omit the NHOLD parameter, this job has no predecessor jobs. JES3 can schedule it for immediate execution.

If the NHOLD count is incorrect, the following can occur:

- If n is greater than the actual number of predecessor jobs, JES3 does not release this job for execution when all of its predecessor jobs complete execution.
- If n is less than the actual number of predecessor jobs, JES3 prematurely releases the job for execution.

NRCMP=HOLD
NRCMP=NOHO
NRCMP=FLSH
Indicates that a network job that completed normally is being resubmitted and that JES3 must erase all references to the job before the job reenters the network.

HOLD
Indicates that JES3 is to hold the job until it is released by the operator.

NOHO
Indicates that JES3 is to allow the job to be scheduled as system resources become available.

FLSH
Indicates that JES3 is to flush the job from the system.

OPHOLD=NO
OPHOLD=YES

NO
Indicates that the job is to be processed normally without operator intervention. If OPHOLD is omitted, NO is the default.

YES
Indicates that JES3 is to hold the job until it is released by the operator.

RELEASE=(jobname[,jobname]...)
Indicates that this job must be executed before the named job(s) in this DJC network can be executed.

jobname
Names the JOB statement for a successor job. You can specify from 1 through 50 successor jobnames.

RELEASE is the only parameter on the //*NET statement that can be split and continued on the next statement. To continue the RELEASE parameter, end the statement with the comma following a jobname and continue the next statement with the next jobname. The left parenthesis appears at the beginning of the jobname list and the right parenthesis appears at the end of the list. For example:

```
//*NET NETID=EXP1,RELEASE=(JOB35,JOB27Z,MYJOB,
//*WRITJB,JOBABC)
```

RELSCHCT=n
Controls early set up of a dependent job's resources. Set up begins when the NHOLD count becomes less than or equal to n. n is a number from 1 through 32767.

If you specify RELSCHCT=0 or omit the RELSCHCT parameter, JES3 does not set up dependent jobs early.

Note: Use this parameter carefully; RELSCHCT can tie up devices and data sets for long times. Do not specify the RELSCHCT parameter:

- For a job that may have catalog dependencies.
- For a job that contains one or more //*PROCESS statements.

Location in the JCL

Place the //*NET statement for a job after the JOB statement and before the first EXEC statement. Code only one //*NET statement for each job in a DJC network.

The //*NET statement must precede any //*PROCESS statements.

Examples of the //*NET Statement

```
//*NET  NETID=NET01,NHOLD=0,DEVPOOL=(,3330,2)
```

This statement defines a DJC network named NET01. The network contains no predecessor jobs. The DEVPOOL parameter, which must be coded in the first job in the network, requests that JES3 establish a device pool of two 3330s for network NET01.

```
//*NET NETID=N1,RELEASE=B,NETREL=(N2,B2)
```

This statement adds a job to the DJC network named N1. This job must be executed before job B, which is in N1, and before job B2, which is in the DJC network named N2.

//*NETACCT Statement

Purpose: Use the //*NETACCT statement to specify accounting information that JES3 is to transmit with a job to another node in the network.

Syntax:

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

The parameters are:

      PNAME=programmer's-name
      ACCT=number
      BLDG=address
      DEPT=dept
      ROOM=room
      USERID=userid
```

- The //*NETACCT statement consists of the characters //* in columns 1 through 3, NETACCT in columns 4 through 10, a blank in column 11, and parameters in columns 9 through 72. JES3 ignores columns 73 through 80.
- Do not continue a //*NETACCT statement. If the parameters cannot fit on one statement, code more than one //*NETACCT statement.
- Enclose any parameter value that contains special characters, including embedded blanks, in apostrophes.

Parameter Definition

PNAME = programmer's-name
Identifies the programmer. programmer's-name is 1 through 20 characters.

ACCT = number
Gives the network account number. number is 1 through 8 characters.

BLDG = address
Gives the programmer's building address. address is 1 through 8 characters.

DEPT = dept
Gives the programmer's department number. dept is 1 through 8 characters.

ROOM = room
Gives the programmer's room number. room is 1 through 8 characters.

USERID = userid
Gives the programmer's network userid. userid is 1 through 8 characters.

Defaults

For any //*NETACCT parameter that is omitted, JES3 uses an installation default specified at JES3 initialization.

Location in the JCL

Place the //*NETACCT statement(s) for a job stream to be transmitted immediately after the first JOB statement and before any //*ROUTE XEQ statements.

Example of the //*NETACCT Statement

```
//*NETACCT     PNAME=COLLINS,ACCT=D58D921,USERID=NXT
```

//*OPERATOR Statement

Purpose: Use the //*OPERATOR statement to issue a message to the operator. Columns 1 through 80 are written on the operator console and in the job's hard-copy log when JES3 reads in the job.

Syntax:

```
//*OPERATOR  message
```

The //*OPERATOR statement consists of the characters //* in columns 1 through 3, OPERATOR in columns 4 through 11, a blank in column 12, and the message for the operator in columns 13 through 80.

Location in the JCL

Place the //*OPERATOR statement anywhere after the JOB statement.

Example of the //*OPERATOR Statement

```
//*OPERATOR CALL EXT. 55523 WHEN THIS JOB STARTS
```

//*PAUSE Statement

Purpose: Use the //*PAUSE statement to halt an input reader temporarily. When you enter a //*PAUSE statement through an input reader, JES3 issues a message and waits for the operator to reply. To start the input reader, the system operator must issue a *START command or a remote work station with console level 15 must send a start message.

The //*PAUSE statement is intended primarily for system checkout and test. It should be issued only by remote work stations.

Syntax:

```
//*PAUSE  [comments]
```

The //*PAUSE statement consists of the characters //* in columns 1 through 3, PAUSE in columns 4 through 8, blanks in columns 9 and 10, and, optionally, comments starting in any column beginning with 11. JES3 ignores columns 73 through 80.

Location in the JCL

Place the //*PAUSE statement before the first JOB statement in an input stream. If it appears after the first JOB statement, JES3 ignores it.

Example of the //*PAUSE Statement

```
//*PAUSE  THIS IS A TEST.
```

//*PROCESS Statement

Purpose: Use the //*PROCESS statement to control how JES3 processes a job. A job that contains //*PROCESS statements receives only the JES3 processing specified on the //*PROCESS statements plus certain required processing.

Specifically, the //*PROCESS statement calls a dynamic support program (DSP) in the DSP dictionary. JES3 must be able to process the called DSP.

Standard Job Processing: JES3 uses a series of processing functions to process a job. Standard processing consists of only the standard scheduler functions:

Converter/interpreter service
Main service
Output service
Purge service

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

Use of Nonstandard Job Processing: Nonstandard job processing is useful in testing. For example, a //*PROCESS statement can make JES3 bypass program execution so that the job's JCL can be checked. Another //*PROCESS statement can make JES3 bypass output processing; then the operator can check by inquiry command whether the job reached execution.

Syntax:

```
//*PROCESS dsp
[parameter[,parameter]...]
```

The //*PROCESS statement consists of the characters //* in columns 1 through 3, PROCESS in columns 4 through 10, a blank in column 11, and the DSP name beginning in column 12. The rest of the columns must be blank.

If the requested DSP requires parameters, code them on the following statement. The parameter statement consists of parameters in columns 1 through 72, separated by commas. Columns 73 through 80 must be blank. Only one parameter statement after a //*PROCESS statement is allowed, any others are ignored by JES3.

Parameter Definition

dsp

Identifies the DSP that JES3 is to use in processing the job. Figure 22-1 lists the valid DSP names and whether parameters can follow.

DSP	DSP Function	Parameters
Standard processing functions:		
CI	JES3 Converter/Interpreter Service, which interprets the JCL and creates control blocks.	Yes (See *JES3 Diagnosis*)
MAIN	Main Service, which processes the program.	No
OUTSERV	Output Service, which processes the job's output.	No
PURGE	Purge Service, which purges the job. This is the last function in any job. JES3 automatically creates this DSP.	No
Nonstandard processing functions:		
CBPRNT	Control Block Print	Yes (See *JES3 Diagnosis*)
DISPDJC	Display Dependent Job Control	Yes (See *JES3 Diagnosis*)
DISPLAY	Display Job Queues	Yes (See *JES3 Diagnosis*)
DJCPROC	Invoke Dependent Job Control Updating *Note:* A //*PROCESS DJCPROC statement is required only when a //*PROCESS MAIN statement is not coded.	No
DR	Disk Reader	Yes (See *JES3 Commands*)
ISDRVR	Input Service Driver (JES3 Control Statement Processing)	Yes (Qualified ddname of input data set)
JESNEWS	Use JESNEWS Facility	Yes (See *JES3 Commands*)
xxx	User-written DSP	(See *SPL: JES3 User Modifications and Macros*)

Figure 22-1. DSPs for JES3 //*PROCESS Statements

Location in the JCL

- Place all //*PROCESS statements for a job immediately after the JOB statement and before the first EXEC statement. If the job includes a //*NET statement, the //*NET statement must appear between the JOB statement and the first //*PROCESS statement.
- The //*PROCESS statements can be separated only by their parameter statements.
- JES3 processes the //*PROCESS statements in the order in which they appear in the input stream.
- The first //*PROCESS statement must request an interpreter DSP if you want input service error messages, which indicate that a job is to be scheduled for interpreter processing before being purged.

Examples of the //*PROCESS Statement

```
//EXAM1  JOB
//*PROCESS  CI
//*PROCESS  MAIN
//*PROCESS  OUTSERV
//S1     EXEC PGM=ANY
         .
         .
         JCL statements
         .
```

This example shows how to submit a simple job via //*PROCESS statements. It is processed like a standard job. The four standard scheduler functions are used for the job: CI, MAIN, OUTSERV, and PURGE. Note that PURGE is not specified; JES3 automatically creates this DSP.

```
//EXAM2  JOB
//*PROCESS  CI
//*PROCESS  MAIN
//*PROCESS  OUTSERV
//*PROCESS  PLOT
//*ENDPROCESS
//S1     EXEC PGM=ANY
//DD1    DD  ...
         .
         .
         JCL statements
         .
```

This example shows how to request a user-written DSP: PLOT. PLOT is to be executed after output service has completed. Note that PURGE is again not specified but is automatically created.

```
//EXAM3  JOB
//*PROCESS  OUTSERV
//*FORMAT   PR,DDNAME=S1.DS1,COPIES=5
//*DATASET  DDNAME=S1.DS1
      .
      .
      data
      .
      .
//*ENDDATASET
//S1  EXEC PGM=ANY
//DS1 DD   DSNAME=DATA1
      .
      .
```

This example uses JES3 output service and the //*DATASET statement. Five copies of data set DS1 are printed on any local printer.

//*ROUTE XEQ Statement

Purpose: Use the //*ROUTE XEQ statement to send the following input stream to a network node where the job is then executed. JES3 stops transmitting input stream records when it finds one of the following:

- The second JOB statement after the //*ROUTE XEQ statement.
- The input stream runs out of card images.

Note: All output from the job is assumed to print/punch at the originating node unless otherwise specified on a DEST parameter.

Syntax:

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

The //*ROUTE XEQ statement consists of the characters //* in columns 1 through 3, ROUTE in columns 4 through 8, a blank in column 9, and, starting in any column from 10 through 72: XEQ, followed by at least one blank and then parameters. JES3 ignores columns 73 through 80.

Do not imbed blanks in the nodename or vmguestid parameters.

Parameter Definition

nodename
Indicates the node. The nodename identifies an MVS JES2 system, an MVS JES3 (global) system, a VSE POWER node, or a VM system.

If nodename specifies a local node:

- The job executes locally if the job begins with a JOB statement.
- The job is terminated if the job begins with an NJE statement.

.vmguestid
Identifies a guest system running in a virtual machine (VM), for example, an MVS system running under VM.

Note: Do not specify a work station or terminal in this parameter.

Location in the JCL

- Place the //*ROUTE XEQ statement after a JOB statement that is valid for the submitting location and any //*NETACCT statements.

- Place the //*ROUTE XEQ statement immediately before a JOB statement that is valid for the executing location.

JES3: //*ROUTE XEQ

JOB Statement after //*ROUTE XEQ

An error in the //*ROUTE XEQ statement can cause the JOB statement following the //*ROUTE XEQ to be processed at the submitting node. To prevent this, code NJB instead of JOB on the second JOB statement; JES3 changes the NJB to JOB before transmitting the job.

Note: TSO users must code NJB instead of JOB on the second JOB statement.

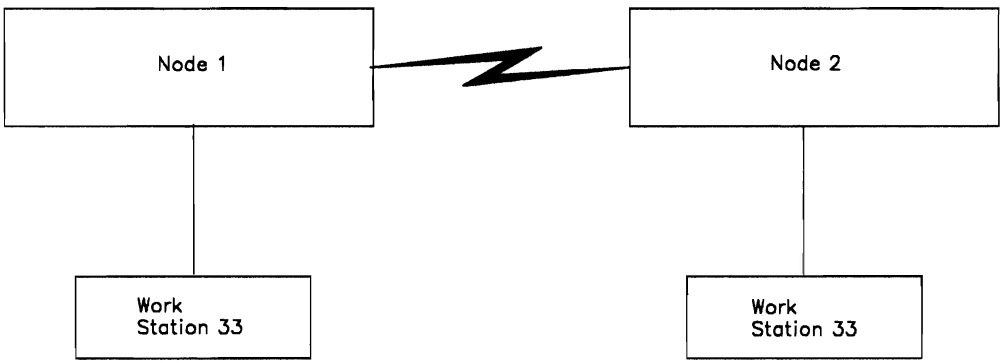
Example of the //*ROUTE XEQ Statement

```
//JOBN1   JOB   options ...
//*ROUTE  XEQ   2
//JOBN2   JOB   options ...
//STEP1   EXEC  PGM=REPORTER
//DD1     DD    SYSOUT=A,DEST=N1R33
//DD2     DD    SYSOUT=A,DEST=N2R33
//DD3     DD    SYSOUT=B,DEST=R33
//DDIN    DD    *
          .
          .
          data
          .
/*
```

In this example, JOB statement JOBN1 is entered through the JES3 system at node 1. The //*ROUTE XEQ statement tells JES3 to send the following input stream to node 2. Transmission of the input stream is stopped by the /* delimiter statement. JOB statement JOBN2 and all following statements until the delimiter are read and executed by the system at node 2.

The sysout data sets are sent to two work stations:

- Sysout data set DD1 is produced at work station 33 attached to node 1.
- Sysout data set DD2 is produced at work station 33 attached to node 2.
- Sysout data set DD3 is produced at work station 33 attached to node 1. Because no node is specified, the originating node is assumed.



/*SIGNOFF Statement

Purpose: Use the /*SIGNOFF statement to tell JES3 to end a remote job stream processing session. At the completion of the current print and/or punch streams, JES3 disconnects the remote work station from the system. If JES3 is reading jobs from the station when the output completes, JES3 disconnects the station when the input is completed.

Both systems network architecture (SNA) and binary synchronous communication (BSC) remote work stations use the /*SIGNOFF statement.

References: For more information on the /*SIGNOFF command, see *SPL: JES3 Initialization and Tuning*.

Syntax:

```
/*SIGNOFF
```

The /*SIGNOFF statement consists of the characters /* in columns 1 and 2, SIGNOFF in columns 3 through 9, and blanks in columns 10 through 80.

Note that, unlike other JES3 statements, this statement starts with only one slash.

Location in the JCL

The /*SIGNOFF statement can appear anywhere in a local input stream or an input stream from a SNA or BSC remote work station.

Example of the /*SIGNOFF Statement

```
/*SIGNOFF
```

This statement requests that JES3 terminate a remote job stream processing session.

/*SIGNON Statement

Purpose: Use the /*SIGNON statement to tell JES3 to begin a remote job stream processing session. The /*SIGNON statement can override the remote identification number normally assigned to the remote work station. This statement is optional for all work stations except non-multi-leaving remote stations on a switched line.

Systems network architecture (SNA) remote work stations must use the LOGON command instead of the /*SIGNON statement to notify JES3 of a connection request.

Syntax:

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

The /*SIGNON statement consists of the following:

Column	Contents
1-2	/*
3-8	SIGNON
9-15	blanks
16-20	work-station-name, beginning in 16
21	blank
22	A or a blank
23	R or a blank
24	blank
25-32	password1, beginning in 25
33-34	blanks
35-42	password2, beginning in 35
43-80	blanks

Note that, unlike other JES3 statements, this statement starts with only one slash.

Parameter Definition

work-station-name
Specifies the name of the remote work station. The work-station-name is 1 through 5 characters and must have been defined on a JES3 RJPTERM initialization statement.

A
Indicates an automatic reader. A can be coded only when the work station is a programmable terminal. Leave this column blank if you do not want to specify an automatic reader.

R
Indicates that print or punch output will be rescheduled if the needed device is not ready. R can be coded only when the work station is a nonprogrammable terminal. Leave this column blank if you do not want to specify the R option.

password1
Specifies the password for the remote job processing (RJP) line. password1 is 1 through 8 characters and must have been defined on a JES3 RJPLINE initialization statement.

password2
Specifies the password for the work station. password2 is 1 through 8 characters and must have been defined on a JES3 RJPTERM initialization statement.

Location in the JCL

Place the /*SIGNON statement at the start of an input stream to be transmitted from a remote work station.

Example of the /*SIGNON Statement

```
/*SIGNON       QUIN  A  PSWD1     PSWD2
```

This statement requests that remote work station QUIN begin a remote job stream processing session. The value **A** in column 22 specifies an automatic reader for the programmable terminal. PSWD1, beginning in column 25, is the password assigned to a dial line. PSWD2, beginning in column 35, is the password assigned to the remote work station.

Index

K

L

MVS/Extended Architecture JCL Reference

GC28-1352-4

S370-36

Printed in U.S.A.

IBM®

MVS/Extended Architecture
JCL Reference

READER'S
COMMENT
FORM

GC28-1352-4

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.)

MVS/Extended Architecture JCL Reference
GC28-1352-4

S370

cut along this line

Reader's Comment Form

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 950
Poughkeepsie, New York 12602-9935

Fold and Tape | Please Do Not Staple | Fold and Tape

Printed in U.S.A.

GC28-1352-04