GC28-0692-5
File No. S370-36

Systems

OS/VS2 MVS JCL

Release 3.8

Includes Selectable Units:

JES3 3850 Mass Storage System	5752-818
JES3 Release 3.0	5752-826

Includes Program Product:

Network Job Entry Facility for JES2 Release 3.1	5840-XR8

IBM

Sixth Edition (April 1984)

This is a reprint of GC28-0692-4 incorporating changes released in the following Technical Newsletters:
GD23-0148-0 (dated 31 May 1979)
GN28-4684-0 (dated 12 November 1979)
GN28-4728-0 (dated 02 May 1980)
GD23-0179-2 (dated 19 December 1980)
GN28-4913-0 (dated 31 March 1981)
GN28-4956-0 (dated 30 December 1981)

This revision incorporates the following Program Products:
MVS/System Product - JES3 5740-XYN
MVS/System Product - JES2 5740-XYS

See the Summary of Amendments following the Contents for a summary of the changes that have been made to this manual.

This edition applies to Release 3.8 of OS/VS2 and to all subsequent releases of OS/VS2 until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *System/370 and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, and services do not imply that IBM intends to make these available in all countries in which IBM operates.

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

A form for readers' comments has been provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, Publications Development, Department D58, Building 706-2, P. O. Box 390, Poughkeepsie, NY 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1974, 1975, 1976, 1977, 1979

About This Book

This publication contains the information necessary to code job control language (JCL) statements, job entry subsystem 2 (JES 2) control statements, and job entry subsystem 3 (JES3) control statements. If you have installed the program product MVS/System Product JES2 Release 2, Program Number 5740-XYS, JES2 information throughout this publication refers to MVS/System Product - JES2 function unless noted otherwise.

The MVS/System Product - JES2 includes the function previously provided by Network Job Entry Facility for JES2 (JES2 NJE) Program Number 5740-XR8 and adds additional function.

If you have installed the program product MVS/System Product - JES3, Release 2 Program Number 5740-XYN, JES3 information throughout this publication refers to MVS/System Product - JES3 function unless noted otherwise.

This book is intended for use by programmers who code JCL, JES2, and JES3 control statements and who understand the concepts of job management and data management.

The book is divided into several chapters:

- The Introduction and the first seven chapters are a guide to using JCL, JES2, and JES3 control statements, written primarily for the inexperienced user of JCL. They contain background information necessary to understand why to code certain parameters, sample situations illustrating when to code these parameters, and descriptions of how to code combinations of parameters to perform particular functions. Examples of jobs involving a variety of parameters are included. The descriptions of JCL services are grouped into seven chapters: Requesting Resources and Identifying Data, Routing a Job Through the System (JES2), Obtaining Output (JES2), Routing a Job Through the System (JES3), Obtaining Output (JES3), Special Data Sets, and Cataloged and In-stream Procedures.
- The next four chapters describe the parameters, their syntax, and rules for coding. The descriptions include the format of each JCL, JES2, and JES3 control statement and the format of the parameters associated with each statement. Parameters for the control statements are presented in alphabetical order giving a brief definition and a reference to the appropriate publication or section for a detailed explanation of the facility or service to be used. The definition and reference are followed by a default, rules for coding, and at least one example of how to code the control statement or parameter. The descriptions are grouped into four chapters: Coding JCL Statements (followed by the JOB, EXEC, and DD statements), Coding JES2 Control Statements, and Coding JES3 Control Statements.
- The last three sections contain reference tables, the glossary, and the index for quick retrieval of information.

JCL Statements no Longer Supported or Supported Differently

A few parameters introduced in OS are no longer supported in MVS. Main storage hierarchy support and the rollout/rollin features are not available in MVS. The system will check the HIERARCHY and ROLL parameters only for correct syntax.

The SEP and AFF parameters and the UNIT=SEP subparameter on the DD statement have no meaning in MVS. If they are coded, they are ignored. However, if coded, they are checked for syntax. The job fails if coded incorrectly.

JCL DD parameters supported differently are SPLIT and SUBALLOC. Their values are internally converted to SPACE requests. When the SUBALLOC keyword is used, the DD statement from which space is allocated becomes a dummy DD. If JES3 is used, the UNIT

parameter on a DD statement which names a cataloged data set cannot specify a device type that conflicts with the cataloged device type (for example, a 3330 and a 2314).

The REGION parameter on the JOB and EXEC statement has the same meaning as in OS/MVT and VS2 Release 1 unless modified by the installation. In virtual storage requests, REGION can be coded to act as an upper limit for variable-length GETMAIN requests.

Prerequisite Publication

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

Publications to which the text refers:

Data Processing Glossary, GC20-1699.

OS/VS2 MVS Checkpoint/Restart, GC26-3877.

OS/VS2 MVS System Programming Library: Data Management, GC26-3830.

OS/VS2 System Programming Library: Debugging Handbook, GBOF-8211.

OS/VS2 System Programming Library: Diagnostic Techniques, GC28-0725.

OS/VS2 System Programming Library: Job Management, GC28-0627.

OS/VS2 Supervisor Services and Macro Instructions, GC28-0683.

OS/VS Tape Labels, GC26-3795.

OS/VS2 MVS Data Management Macro Instructions, GC26-3873.

Operator's Library: OS/VS2 MVS System Commands, GC28-1031.

OS/VS2 MVS System Programming Library: VTAM, GC28-0688.

OS/VS2 MVS System Programming Library: Initialization and Tuning Guide, GC28-1029.

OS/VS2 MVS Data Management Services Guide, GC26-3875.

OS/VS2 Access Method Services, GC26-3841.

OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide, GC26-3838.

OS/VS2 MVS Utilities, GC26-3902.

OS/VS2 MVS System Programming Library: System Management Facilities (SMF), GC28-1030.

OS and OS/VS Programming Support for the IBM 3505 Card Reader and IBM 3525 Card Punch, GC21-5097.

OS/VS2 Using OS Catalog Management with the Master Catalog: CVOL Processor, GC35-0010.

OS/VS2 TCAM System Programmer's Guide, (levels 8 and 9), GC30-2041.

OS/VS2 TCAM System Programmer's Guide, (level 10), GC30-2051.

OS/VS BTAM, GC27-6980.

Graphic Programming Services for 2250, GC27-6971.

Graphic Programming Services for 2260, GC27-6972.

IBM 3340 Fixed Head Feature Users Guide, GA26-1632.

IBM 2821 Control Unit, GA24-3312.

IBM 3800 Printing Subsystem Programmer's Guide, GC26-3846.

Forms Design Reference Guide for the IBM 3800 Printing Subsystem, GA26-1633.

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

OS/VS Mass Storage System (MSS) Services: Reference Information, GC35-0017.

OS/VS2 System Programming Library: System Generation Reference, GC26-3792.

OS/VS2 System Programming Library: Service Aids, GC28-0674.

OS/VS2 System Programming Library: Supervisor, GC28-1046.

OS/VS2 IBM 3540 Programmer's Reference, GC24-5111.

OS/VS2 TSO Command Language Reference, GC28-0646.

System Programming Library: JES2 Installation, Initialization, and Tuning, SC23-0046.

Operator's Library: JES2 Commands, SC23-0048.

OS/VS2 MVS Resource Access Control Facility (RACF) General Information Manual, GC28-0722.

OS/VS2 MVS Interactive Problem Control System (IPCS) System Information, GC34-2004.

JES3 System Programming Library: Installation Planning and Tuning, SC23-0041.

JES3 System Programming Library: User Modifications and Macros, SC23-0042.

JES3 System Programming Library: Diagnosis, SC23-0043.

JES3 Operator's Library, SC23-0045.

Contents

About This Book . . . iii
JCL Statements no Longer Supported or Supported Differently . . . iii
Prerequisite Publication . . . iv
Publications to which the text refers . . . iv

Summary of Amendments . . . xvii

Introduction . . . 1
The JCL Statements . . . 1
The JES2 Statements (5740-XYS) . . . 4
The JES3 Statements . . . 5
Cataloged and In-stream Procedures . . . 6
Processing Your Job . . . 6

Requesting Resources and Identifying Data . . . 9
Requesting Storage for Execution of a Program . . . 9
When to Request Real Storage . . . 9
Specifying Storage Requirements with the REGION Parameter . . . 10
Example of Requesting Storage . . . 10
Requesting Units and Volumes . . . 10
Specifying Volume Information . . . 11
Specific Volume Requests . . . 11
Nonspecific Volume Requests . . . 11
Using Private Volumes . . . 12
Sharing Volumes Between Data Sets . . . 13
Multivolume Data Sets . . . 13
Specifying Unit Information . . . 13
Requesting More than One Unit . . . 14
Deferred Mounting of Volumes . . . 14
When You Do Not Have to Code the UNIT Parameter . . . 15
Sharing a Unit Between Data Sets on Different Volumes . . . 15
Example of Requesting Units and Volumes . . . 19
Requesting Space for Non-VSAM Data Sets . . . 20
The Basic Request: Unit of Measurement and Primary Quantity . . . 20
How the System Satisfies Your Primary and Secondary Requests . . . 21
A Secondary Request for Space . . . 21
Requesting Directory Space for a Partitioned Data Set . . . 22
Requesting Index Space for an Indexed Sequential Data Set . . . 22
Assigning Specific Tracks . . . 22
Example of Requesting Space . . . 23
Mass Storage System (MSS) Considerations . . . 24
Mass Storage Volume Groups . . . 24
Nonspecific Volume Requests for Mass Storage Volumes . . . 24
Specific Volume Requests for Mass Storage Volumes . . . 25
Requesting Space for Non-VSAM Data Sets on Mass Storage Volumes . . . 25
Dynamically Allocating and Deallocating Data Sets . . . 25
Example of Dynamically Deallocating Data Sets . . . 26
Identifying Data Sets to the System . . . 27
Specifying the DDNAME Parameter . . . 27
When You Code the DDNAME Parameter . . . 27
Specifying the DSNAME Parameter . . . 28
Creating or Retrieving a Nontemporary Data Set . . . 28
Nontemporary Data Sets . . . 28
Members of a Partitioned Data Set . . . 29
Generations of a Generation Data Group . . . 29
Areas of an Indexed Sequential Data Set . . . 29
Creating or Retrieving a Temporary Data Set . . . 29
Temporary Data Sets . . . 30
Members of a Temporary Partitioned Data Set . . . 30
Areas of a Temporary Indexed Sequential Data Set . . . 30
Associated Data Sets (3540 Diskette) . . . 31
Copying the Data Set Name From an Earlier DD Statement . . . 31
Specifying the DSNAME Parameter in Apostrophes . . . 31

The Label Type Subparameter 32
The PASSWORD and NOPWREAD Subparameters 34
The IN and OUT Subparameters 34
The RETPD and EXPDT Subparameters 34
Example of Identifying Data Sets to the System 35
Disposition Processing of Non-VSAM Data Sets 35
Specifying Data Set Status 36
Specifying a Disposition for the Data Set 36
Deleting a Data Set 37
Keeping a Data Set 37
Cataloging a Data Set 38
Uncataloging a Data Set 38
Passing a Data Set 38
Disposition Processing of Passed Data Sets 39
Default Disposition Processing 40
Bypassing Disposition Processing 40
Insuring Data Set Integrity 40
Exclusive Control of a Data Set 40
Shared Control of a Data Set 41
How the System Performs Data Set Integrity Processing 41
Example of Disposition Processing of Non-VSAM Data Sets 42

Routing a Job Through the System (JES2) (5740-XYS) 43
Job Scheduling 43
Assigning a Job to a Job Class 43
Assigning a Priority to a Job 44
Assigning a Dispatching Priority to Job Steps 44
Performance of Jobs and Job Steps 44
Routing a Job in a Network (JES2) **(5740-XYS)** 45
Transmitting Data in a Network (JES2) **(5740-XYS)** 45
Passing Information to the Job in Execution 45
Identifying the Program to be Executed 45
Temporary Library 45
Private Library 46
The IEFBR14 Program 46
Selecting a Cataloged Procedure Library 46
Delaying Job Initiation 46
Bypassing Job Initiation 47
Conditional Execution of Job Steps 47
Specifying Return Code Tests 48
Restarting a Job 48
Example of Routing a Job Through the System 49

Obtaining Output (JES2) (5740-XYS) 51
Requesting Listings of JCL Statements and System Messages 51
Requesting an Abnormal Termination Dump 51
Writing Output Data Sets 52
Assigning Output Data Sets to Output Classes 52.1
Specifying the Device 53
Processing Output Classes 53
Delaying the Writing of an Output Data Set 54
Suppressing the Writing of an Output Data Set 54
Limiting Output Records 54
Requesting Page Overflow Processing 54
Interpretation of Punched Output 55
JES2 Support of the 3211 Indexing Feature 55
Requesting Multiple Copies of an Output Data Set 55
Requesting Copy Modification 56
Requesting Printer Form and Character Control 56
Requesting a Special Output Form 56
Requesting a Special Character Set Using the UCS Feature 56
Requesting Character Arrangements with a 3800 Printing Subsystem 57
Requesting Forms Control 57
Requesting Forms Overlay 58
Bursting of Output 58
Controlling Output Destination (JES2) **(5740-XYS)** 59
Example of Obtaining Output 60

Routing a Job Through the System (JES3) 61
Scheduling a Job 61

Selecting a Processor . . . 62

Routing a JOB in a Network (JES3 networking) **(5740-XYN)** . . . 62
Allocating Data Resources . . . 63
Spool Partitioning **(5740-XYN)** . . . 63
Types of JES3 Setup . . . 63
Selecting a Job . . . 66
Deadline Scheduling . . . 67
Postponing Job Selection . . . 67
Performance of Jobs and Job Steps . . . 67
Assigning a Dispatching Priority to Job Steps . . . 68
Dependent Job Control . . . 68
How to Code NET Statements . . . 70
Examples of Dependent Job Control . . . 70
Conditional Execution of Job Steps . . . 73
Specifying Return Code Tests . . . 74
Passing Information to the Job in Execution . . . 74
Identifying the Program to be Executed . . . 74
Temporary Library . . . 74
Private Library . . . 74
The IEFBR14 Program . . . 75
Testing JCL Without Execution . . . 75
Selecting a Cataloged Procedure Library . . . 75
Reading Column Binary Input . . . 75
Restarting a Job . . . 76
Example of Routing a Job Through the System (JES3) . . . 77
Example of Routing a Job Through a Network (JES3) **(5740-XYN)** . . . 77

Obtaining Output (JES3) . . . 79
Requesting Listings of JCL Statements and System Messages . . . 79
Requesting an Abnormal Termination Dump . . . 79
Writing Output Data Sets . . . 80
Assigning Output Data Sets to Output Classes . . . 80
Specifying the Device . . . 81
Processing Output Classes . . . 81
Delaying the Writing of an Output Data Set . . . 81
Suppressing the Writing of an Output Data Set . . . 82
Limiting Output Records . . . 82
Requesting Multiple Copies of an Output Data Set . . . 83
Requesting Copy Modification . . . 83
Requesting Printer Form and Character Control . . . 83
Requesting a Special Output Form . . . 83
Requesting a Special Character Set Using the UCS Feature . . . 84
Requesting Character Arrangements with a 3800 Printing Subsystem . . . 84
Requesting Forms Control . . . 85
Requesting Forms Overflow and Printer Spacing . . . 85
Requesting Punch Output Interpretation on a 3525 . . . 86
Requesting Forms Overlay . . . 86
Bursting of Output . . . 86
Controlling Output Destination . . . 86
Remote Job Processing . . . 87
Controlling Job and Output Destination Using JES3 Networking **(5740-XYN)** . . . 87
Example of Obtaining Output (JES3) . . . 88

Special Data Sets . . . 89
Creating and Using Private and Temporary Libraries . . . 89
Creating a Private Library . . . 89
Adding Members to a Private Library . . . 90
Retrieving an Existing Private Library . . . 91
Concatenating Private Libraries . . . 92
Using Private Catalogs . . . 92
Temporary Libraries . . . 92
Defining a Dummy Data Set . . . 93
Coding the DUMMY Parameter . . . 93
Coding DSNAME=NULLFILE . . . 93
Requests to Read or Write a Dummy Data Set . . . 94
Using Virtual Input/Output (VIO) for Temporary Data Sets . . . 94
Defining a VIO Temporary Data Set . . . 94
Backward References for VIO Data Sets . . . 94
Using Virtual Input/Output (VIO) to Pass Temporary Data Sets Among Job Steps . . . 95
Entering Data Through the Input Stream . . . 96
VSAM Data Sets . . . 96

VSAM Data Sets . . . 96

Creating and Retrieving Indexed Sequential Data Sets . . . 100
Creating an Indexed Sequential Data Set . . . 100
The DSNAME Parameter . . . 100
The UNIT Parameter . . . 101
The VOLUME Parameter . . . 101
The LABEL Parameter . . . 101
The DCB Parameter . . . 101
The DISP Parameter . . . 102
The SPACE Parameter . . . 102
Nonspecific Allocation Technique . . . 102
Absolute Track Technique . . . 102
Area Arrangement of an Indexed Sequential Data Set . . . 103
Retrieving an Indexed Sequential Data Set . . . 103
The DSNAME Parameter . . . 103
The UNIT Parameter . . . 103
The VOLUME Parameter . . . 104
The DCB Parameter . . . 104
The DISP Parameter . . . 104
Examples of Creating and Retrieving an Indexed Sequential Data Set . . . 104
Creating and Retrieving Generation Data Sets . . . 105
Building a Generation Data Group Base Entry . . . 105
Creating a Model Data Set Label . . . 105
Referring the System to a Cataloged Data Set . . . 106
Creating a Generation Data Set . . . 106
The DSNAME Parameter . . . 106
The DISP Parameter . . . 106
The UNIT Parameter . . . 106
The VOLUME Parameter . . . 106
The SPACE Parameter . . . 106
The LABEL Parameter . . . 107
The DCB Parameter . . . 107
Retrieving a Generation Data Set . . . 107
The DSNAME Parameter . . . 107
The DISP Parameter . . . 108
The UNIT Parameter . . . 108
The VOLUME Parameter . . . 108
The LABEL Parameter . . . 108
The DCB Parameter . . . 108
Submitting a Job for Restart . . . 108
Example of Creating and Retrieving Generation Data Sets . . . 109
Creating and Using a Subsystem Data Set . . . 109

Cataloged and In-Stream Procedures . . . 110
Writing Cataloged and In-Stream Procedures . . . 110
Identifying an In-stream Procedure . . . 110
Placing a Cataloged Procedure in a Procedure Library . . . 111
Allowing for Changes in Cataloged and In-stream Procedures . . . 111
Using Cataloged and In-Stream Procedures . . . 112
How to Call Cataloged and In-Stream Procedures . . . 112
Modifying Cataloged and In-Stream Procedures . . . 112
Modifying Parameters on an EXEC Statement . . . 112
Modifying Parameters on a DD Statement . . . 114
Modifying Parameters on DD Statements that Define Concatenated Data Sets . . . 115
Adding DD Statements to a Procedure . . . 116
Identifying Procedure Statements on an Output Listing . . . 116
Using Symbolic Parameters . . . 117
Defining Symbolic Parameters When Writing a Procedure . . . 117
Assigning Default Values to Symbolic Parameters . . . 118
Assigning Values to and Nullifying Symbolic Parameters . . . 118
Assigning a Value to a Symbolic Parameter . . . 119
Nullifying a Symbolic Parameter . . . 120
Caution Concerning Leading and Trailing Commas . . . 120
Example of a Procedure Containing Symbolic Parameters . . . 122

Coding JCL Statements . . . 123
Notation for Defining Control Statement Parameters . . . 123
Fields in JCL Control Statements . . . 124
Parameters in the Operand Field . . . 125
Continuing Control Statements . . . 126
Concatenating Data Sets . . . 126

Using Special Characters 127

The JOB Statement 129
The Accounting Information Parameter 130
The ADDRSPC Parameter 132
The CLASS Parameter 133
The COND Parameter 134
The GROUP Parameter 135
The MSGCLASS Parameter 136
The MSGLEVEL Parameter 137
The NOTIFY Parameter 138
The PASSWORD Parameter 139
The PERFORM Parameter 140
The Programmer's Name Parameter 141
The PRTY Parameter (JES2) 142
The PRTY Parameter (JES3) 143
The RD Parameter 144
The REGION Parameter 146
The RESTART Parameter 147
The TIME Parameter 149
The TYPRUN Parameter 151
The USER Parameter 153

The EXEC Statement 155
The ACCT Parameter 156
The ADDRSPC Parameter 157
The COND Parameter 158
The DPRTY Parameter 161
The DYNAMNBR Parameter 162
The PARM Parameter 163
The PERFORM Parameter 165
The PGM Parameter 166
The PROC Parameter 168
The RD Parameter 169
The REGION Parameter 171
The TIME Parameter 172

The DD Statement 175
The JOBCAT Facility 177
The JOBLIB Facility 178
The STEPCAT Facility 181
The STEPLIB Facility 182
The SYSABEND, SYSMDUMP, and SYSUDUMP Facilities 185
The SYSCHK Facility 188
The SYSCKEOV Facility 190
The * Parameter 191
The AMP Parameter 193
The BURST Parameter 196
The CHARS Parameter 197
The CHKPT Parameter 198
The COPIES Parameter 199
The DATA Parameter 201
The DCB Parameter 203
The DDNAME Parameter 217
The DEST Parameter 219
The DISP Parameter 221
The DLM Parameter 224
The DSID Parameter 225
The DSNAME Parameter 226
The DUMMY Parameter 229
The DYNAM Parameter 231
The FCB Parameter 232
The FLASH Parameter 234
The FREE Parameter 235
The HOLD Parameter 237
The LABEL Parameter 238
The MODIFY Parameter 242
The MSVGP Parameter 243
The OUTLIM Parameter 245
The PROTECT Parameter 246

The QNAME Parameter 247

The SPACE Parameter 248
The SUBSYS Parameter 252
The SYSOUT Parameter 254
The TERM Parameter 256
The UCS Parameter 257
The UNIT Parameter 259
The VOLUME Parameter 262

The Command Statement 266

The Comment Statement 267

The Delimiter Statement 269

The Null Statement 271

The PEND Statement 273

The PROC Statement 275

Coding JES2 Control Statements (5740-XYS) 277
The Command Statement 278
The JOBPARM Statement 279
The MESSAGE Statement 281
The NETACCT Statement 281.0
The NOTIFY Statement 281.1
The OUTPUT Statement 282
The PRIORITY Statement 285
The ROUTE Statement 286
The SETUP Statement 288
The SIGNOFF Statement 289
The SIGNON Statement 290
The XEQ Statement 290.1
The XMIT Statement 290.2

Coding JES3 Control Statements (5740-XYN) 291
The Command Statement 292
The DATASET Statement 294
The ENDDATASET Statement 296
The ENDPROCESS Statement 297
The FORMAT Statement 298
The PR Parameter 303
The PU Parameter 309
The MAIN Statement 312
The NET Statement 320
The NETACCT Statement 323.0
The OPERATOR Statement 324
The PAUSE Statement 325
The PROCESS Statement 326
The ROUTE Statement 328

Reference Tables 349

Glossary 365

Index 371

Figures

Figure 1. Job Control Statements 1
Figure 2. Deleted by **(5740-XYS)** 3
Figure 3. JES2 Control Statements **(5740-XYS)** 4
Figure 4. JES3 Control Statements **(5740-XYN)** 5
Figure 5. A Job in the Input Stream 7
Figure 6. A Job with Several Job Steps 8
Figure 7. Job Boundaries in the Input Stream 8

Figure 8. How Device Status Affects Eligibility for Allocation 11

Figure 9. Unit and Volume Affinity 17
Figure 10. Special Character Sets for the 1403 and 3211 Printers (JES2) 57
Figure 11. Example of JES3 Setup 65
Figure 12. Special Character Sets for the 1403 and 3211 Printers (JES3) 84
Figure 13. DD Parameters used with VSAM 97
Figure 14. DD Parameters you should avoid with VSAM (Part 1 of 2) 98
Figure 14. DD Parameters you should avoid with VSAM (Part 2 of 2) 99
Figure 15. Identification of Cataloged Procedure Statements on the Output Listing 117
Figure 16. Identification of In-stream Procedure Statements on the Output Listing 117
Figure 17. JCL Control Statement Fields 124
Figure 18. Character Sets 127
Figure 19. DD Parameters for Creating a Data Set (Part 1 of 2) 350
Figure 19. DD Parameters for Creating a Data Set (Part 2 of 2) 351
Figure 20. DD Parameters for Retrieving a Data Set 352
Figure 21. DD Parameters for Extending a Data Set 353
Figure 22. DD Parameters for Retrieving or Extending an Indexed Sequential Data Set 354
Figure 23. Area Arrangement of Indexed Sequential Data Sets 355
Figure 24. Table of Mutually Exclusive DD Parameters 356
Figure 25. Disposition Processing Chart 357
Figure 26. Direct Access Capacities 358
Figure 27. Track Capacities (Part 1 of 2) 358.1
Figure 27. Track Capacities (Part 2 of 2) 358.2
Figure 28. Table of Allowable DSPs for PROCESS Statements 359
Figure 29. The JOB Statement 360
Figure 30. The EXEC Statement 361
Figure 31. The DD Statement (Part 1 of 3) 362
Figure 31. The DD Statement (Part 2 of 3) 363
Figure 31. The DD Statement (Part 3 of 3) 364

The JOB Statement ——— JOB

The EXEC Statement ——— EXEC

The DD Statement ——— DD

Miscellaneous JCL Statements ——— MISC JCL

The JES2 Statements ——— JES2

The JES3 Statements ——— JES3

(Removed by 5740-XYS)

Reference Tables ——— REF TAB

Glossary ——— GLOS-SARY

Summary of Amendments for GC28-0692-4 as updated by GN28-4728

Changes have been made throughout this publication to correct technical inaccuracies and clarify technical ambiguities. Some of the areas affected are:

- Retrieving a Generation Data Set
- Cataloging of data sets created by the IEFBR14 Program
- Handling of duplicate ddnames in a job step
- Additional rules for coding the RLSE and ROUND subparameters of the label parameter

Additionally, new information describes how to retain the first SYSMDUMP on a data set and optionally to record additional SYSMDUMPS to the same data set.

Summary of Amendments for GC28-0692-4 as updated by GN28-4684

Changes have been made throughout this publication to reflect technical changes and to include information for the following:

- Documentation to support the 3203-5 printer.
- Documentation of the Interactive Problem Control System (IPCS).
- Documentation of miscellaneous technical changes.

Summary of Amendments for GC28-0692-4 OS/VS2 Release 3.8

Changes have been made throughout this publication to reflect technical changes and to include information for the following:

JES3 Release 3.0, SU26, GD23-0098

The following information was added, which defines how to code JCL statements and JES3 control statements for System Network Architecture (SNA) Remote Job Processing (RJP):

- Definition of the values you can specify on a DD * statement for JES3 SNA RJP input devices.
- An explanation of how to code the DD FCB parameter for work stations that use a peripheral data set information record (PDIR) and for work stations that do not.
- Addition of two subparameters (CHNSIZE and COMPACT) to the JES3 PR and PU parameters.

Network Job Entry Facility for JES2 Release 2.0 Program Number 5740-XR8, SC28-0786

The following information was added:

- Descriptions of additional subparameters that can be coded in the DEST parameter.
- Descriptions of the following NJE statements:
  NETACCT
  NOTIFY
  XEQ.

Summary of Amendments for GC28-0692-3 OS/VS2 Release 3.7 and Listed SUs

Changes have made throughout this publication to reflect a Service Update to OS/VS2 Release 3.7 and to include information for the listed Selectable Units.

JES2 Release 4.0, SU3, GN28-2648

Information was added for JES2 Release 4.0 in the following areas:

- On the JOB statement:
  - JES2 parameters were added to the Accounting Information parameter.
  - The PRTY parameter was added.
  - The TYPRUN subparameter was added to the JCLHOLD parameter.
- On the DD statement, subparameters were added to the DEST parameter.
- On the JES2 JOBPARM statement, the RESTART parameter was added.
- On the JES2 OUTPUT statement, the COPYG, BURST, CHARS, FLASHC, MODIFY, and MODTRC parameters were added to support the IBM 3800 Printing Subsystem.
- The JES2 SIGNOFF and SIGNON statements were added.

Scheduler Improvements, SU4, GN28-2703

Information was added for Scheduler Improvements in the following areas:

- On the JOB statement, the GROUP, PASSWORD, and USER parameters were added for RACF-defined users.
- On the DD statement, the RETAIN subparameter was added to the VOLUME parameter.

Supervisor Performance #2, SU7, GN28-2702

Information was added for Supervisor Performance #2 in the following areas:

- On the EXEC statement, the default for the DPRTY parameter was modified.
- The SETDMN and PAGEADD operator commands were added.

IBM 3800 Printing Subsystem, SU10, GN28-2711

Information was added for the 3800 Printing Subsystem in the following areas:

- On the DD statement, the BURST, CHARS, FLASH, and MODIFY parameters were added.
- On the DD statement, the COPIES and DCB parameters were modified.
- On the JES3 FORMAT statement, the CHARS, FLASH, MODIFY, and STACKER subparameters were added to the PR parameter.

JES3 3850 Mass Storage System, SU18, GC28-0806-0

Information was added for JES3 3850 Mass Storage System in the following areas:

- On the JES3 MAIN statement, the MSS parameter was added.
- On the JES3 NET statement, the SDGxx subparameter was added to the DEVPOOL parameter.

MSS Enhancements, SU24, GC28-0789-0

Information was added for MSS Enhancements in the following area:

- On the DD statement, the ddname operand was added to the MSVGP parameter.

JES2 Release 4.1, SU25, GC28-0799-0

Information was added for JES2 Release 4.1 in the following area:

- On the JES2 OUTPUT statement, the Z=nn parameter was added.

3838 Vector Processing Subsystem Support, SU29, GC28-0927

Information was added for 3838 Vector Processing Subsystem Support in the following area:

- On the DD statement, the SUBSYS parameter was added.

System Security Support, SU32, GC28-0844-0

Information was added for System Security Support in the following area:

- On the DD statement, the PROTECT parameter was added.

Dumping Improvements, SU33, GC28-0820-0

Information was added for Dumping Improvements in the following area:

- The SYSMDUMP DD statement was added.

IBM 3800 Printing Subsystem 12 Lines per Inch, SU48, GN28-2773

Information was added for 3800 Printing Subsystem 12 Lines per Inch in the following area:

- The list of character arrangement tables supplied with the 3800 Printing Subsystem was modified.

Summary of Amendments for GC28-0692-2 OS/VS2 Release 3.7

Changes have been made throughout this publication to reflect a Service Update to OS/VS2 Release 3.7. In addition pertinent technical and editorial changes have been made.

PROCESS and ENDPROCESS Statements (JES3)

These statements, for nonstandard job processing, are now described in this manual.

DD Statement under JES3

Job terminates if duplicate ddnames exist in a step; reserved ddnames are included.

DATASET Statement (JES3)

Additional rules for coding.

FORMAT Statement (JES3)

AC DEST parameter rewritten; rules for coding DDNAME and DEST are clarified.

LABEL Parameter

Syntax and rules for coding are clarified.

TIME Parameter

Relationship of TIME on the JOB and the EXEC statement is clarified.

UNIT Parameter

P subparameter and rules for coding AFF are clarified.

VOLUME Parameter

PRIVATE subparameter is rewritten; RETAIN subparameter is now supported to ensure that a volume remains mounted until the end of the job; rules for coding are clarified.

AMP Parameter

The AMORG and BUFSP subparameters are clarified.

DCB Parameter

Additional information for completing a data control block; descriptions of several subparameters are expanded.

PARM Parameter

Continuation of expressions on another statement is clarified.

Direct Access Devices

Tables updated to reflect the addition of the 3344 and 3350 storage devices.

Command Statement

All JCL command statements are ignored by JES3.

DEST Parameter

REMOTEn is not a valid subparameter for JES2.

Region Size Requirements

For ADDRSPC=VIRT, two internal values are established to limit all GETMAINs.

Creating an ISAM Data Set

If a nonspecific volume request is made, and space is not available on the volume selected, the job fails.

You can write programs in any one of a number of languages. The operating system will translate the language into machine language so that the instructions can be executed and the work performed. There is also a language, called **job control language** (JCL), that directs the operating system in the handling of application programs. When submitting programs to the operating system, you can provide JCL statements to define the work to be done, the methods to be used, and the resources needed. In addition, you can obtain special input and output processing by including JES2 (job entry subsystem 2) and JES3 (job entry subsystem 3) control statements for the job entry subsystems. A collection of related problem programs is submitted to the operating system as a **job**. A job is made up of one or more **job steps**, each of which is a unit of work associated with the overall processing program.

The JCL Statements

Job control language consists of nine statements. The name and purpose of each statement is summarized in Figure 1.

Every job requires the use of the JOB statement (to identify the job), EXEC statements (to identify each job step), and DD statements (to identify data sets used by the job). The null statement is optional. Placing it within the job causes JCL statements other than the JOB statement behind the null statement to be ignored. JES2 ignores the null statement. The delimiter statement can be used to indicate the end of data in the input stream. PROC and PEND statements are used to define a set of JCL statements to be used as an in-stream procedure. The command statement allows operator commands to be submitted through the input stream; this statement is used primarily by the operator. The comment statement can be used to make the programs readily understandable by other programmers and by yourself.

Name of Statement	Purpose
// JOB (job)	marks the beginning of a job; assigns a name to the job.
// EXEC (execute)	marks the beginning of a job step; identifies the programs to be executed or the cataloged or in-stream procedure to be called; assigns a name to the step.
// DD (data definition)	identifies a data set and describes its attributes.
/* (or two characters designated by the user to indicate delimiter)	indicates the end of data placed in the input stream.
// (null)	marks the end of a job.
// PROC (procedure)	for cataloged procedures, assigns default values to parameters defined in the procedure; for in-stream procedures, marks the beginning of the procedure.
// PEND (procedure end)	marks the end of in-stream procedure.
//* (comment)	contains comments.
// (command)	enters system operator commands through the input stream.

Figure 1. Job Control Statements

In addition to identifying data sets, job steps, and the job, you can code parameters on JCL statements to request resources and services from the operating system. The operating system, together with your job entry subsystem, is responsible for managing all the resources of the computing system. It automatically performs many services in processing jobs; however, you can influence the processing of a job by including JCL parameters. For example, JES2 selects a job for execution, but you can influence when the job is selected or delay its selection by coding parameters on the JOB statement (or on the MAIN statement for JES3 processed jobs).

You can also ask for a specific volume on which to write a data set. The following paragraphs describe some of the functions that are available on the major JCL statements:

JOB statement: By using the parameters allowed on the **JOB** statement, you can provide accounting information for the installation's accounting routines, define execution characteristics, specify conditions for early termination of the job, request a specific class for job scheduler messages, hold a job for later execution, and limit the maximum amount of time the job can use the central processing unit (CPU).

EXEC statement: Parameters on the EXEC statement can define the program or cataloged procedure that the system is to execute. They can also be used to provide job step accounting information, to give conditions for bypassing or executing a job step, to assign a limit on the CPU time used by a job step, and to pass information to a processing program such as the linkage editor.

DD statement: Parameters on the DD statement provide the system with such information as the name of the data set, the name of the volume on which it resides, the type of I/O device that holds the data set, the format of the records in the data set, whether a data set is old or new, the size of newly created data sets, and the access method that will be used to create or refer to the data.

Figure 2 deleted by Suppl. GD23-0179-2 for 5740-XYS

The JES2 Statements (5740-XYS)

You can control the input, output, and processing of a program by coding JES2 control statements and placing them in the input stream. The name and function of the JES2 statements that you can use with JCL to direct the execution of a program are shown in Figure 3.

Name of Statement	Purpose
/*$command	enters JES2 operator commands through the input stream.
/*JOBPARM	indicates job related parameters that can be specified at input time.
/*MESSAGE	sends messages to the operator via the operator console.
/*NETACCT	specifies an account number for a network job.
/*NOTIFY	specifies the destination of notification messages.
/*OUTPUT	specifies characteristics and options of a specific SYSOUT data set or groups of SYSOUT data sets.
/*PRIORITY	assigns a job selection priority.
/*ROUTE	specifies the default output destination.
/*XEQ	specifies the execution node for a job.
/*SETUP	indicates volumes needed for executing your job.
/*SIGNOFF	indicates termination of a remote session.
/*SIGNON	indicates the start of a remote session.
/*XMIT	indicates a job or data to be transmitted to another JES2 or eligible non-JES2 node.

Figure 3. JES2 Control Statements

Use JES2 control statements to request more efficient use of resources. Five of these statements contain specific functions that are discussed below.

JOBPARM statement: Parameters on the JOBPARM statement specify the estimated number of cards to be produced as output from a job, the number of copies of printed output desired, the default print or punch forms, the number of output lines on each page, the estimated total number of output lines from the job, your room number, any system affinity that may be required, the estimated job execution time, the printing of the JES2 job log, and the name of the cataloged procedure library to be used to convert the JCL for the job.

NETACCT statement: The network account number enables the user to specify accounting information that can be accepted and interpreted by all nodes as desired. One node might use the account number as is, while another will have the network account number translated to a local account number.

OUTPUT statement: Parameters on the OUTPUT statement specify the characteristics of a SYSOUT data set such as the number of copies if each data set desired, the destination device of the output, any special forms required, the indexing print position offset (3211 only) and forms control buffer image (FCB), and the use of the universal character set (UCS).

ROUTE statement: Parameters on the ROUTE statement direct the execution of a job to any central processor in the network and direct the printed or punched output to any local device, remote terminal, or remote device in the network.

XEQ Statement: Routes the execution of a job to any central processor in the JES2 network.

XMIT Statement: Transmitts a data stream or job to specified JES2 or eligible non-JES2 nodes without JES2 input service functions being performed on the data stream.

The JES3 Statements (5740-XYN)

You can also control the input, output, and processing of a program by coding JES3 control statements and placing them in the input stream. The name and function of the JES3 statements that you can use with JCL to direct the execution of a program are shown in Figure 4.

Name of Statement	Purpose
//**command	enters JES3 operator commands, except *DUMP and *RETURN, through the input stream.
//*DATASET	permits additional input data sets from the input stream.
//*ENDDATASET	terminates the creation of an input data set.
//*ENDPROCESS	terminates a series of PROCESS statements.
//*FORMAT	specifies special destination and format related instructions for a specific SYSOUT or JES3-managed print and punch data set.
//*MAIN	defines selected processing parameters for the current job.
//*NET	identifies relationships between predecessor and successor jobs in a dependent job control net.
//*NETACCT	specifies an account number for network processing.
//*OPERATOR	transmits messages to the operator.
//**PAUSE	halts the input reader.
//*PROCESS	identifies a nonstandard job.
//*ROUTE	specifies the destination node in a network.

Figure 4. JES3 Control Statements

Several of the JES3 statements contain keyword parameters that define options available to help improve the processing of the job. Three of these statements are briefly defined here.

FORMAT statement: Parameters on the FORMAT statement differ according to the type of request you are making. For print and punch data sets, keyword parameters specify such options as output destination, number of output copies, and types of output forms.

MAIN statement: Parameters on the MAIN statement specify such options as the main processor name or type of system to be used for the job, the type of control program to be used, the estimated number of cards or lines of output, the job class for the job, and the time that the job is due to be completed.

ROUTE statement: Parameters on the ROUTE statement direct a job to another node in the network for execution.

Nonstandard job processing. A job, in addition to being a collection of related problem programs identified by a JOB statement, is also a collection of JES3 processing segments, called job segments. A job that consists only of a collection of related problem programs to be processed by MVS and that requires no special job segments is called a standard job. A nonstandard job requires one or more special job segments in place of or in addition to the standard job segments of interpreter service, main service, and output service. Specify a nonstandard job by following the JOB statement with a JES3 PROCESS statement for each job segment.

Cataloged and In-Stream Procedures

Often the same set of JCL statements are used repeatedly with little or no change (for example, to specify compilation, link-editing, and execution of programs). To save programming time and to reduce the possibility of error, standard job step definitions can be prepared and placed (or cataloged) in a partitioned data set known as the procedure library. A set of JCL statements placed in the procedure library is called a cataloged procedure. A cataloged procedure consists of EXEC and DD statements. Up to 255 job steps are permitted in a job including all steps in any procedure called by that job. Specification of a greater number of steps produces unpredictable results.

By simply using a JOB statement and an EXEC statement, you can retrieve a specific cataloged procedure. Specify the name of the procedure on the EXEC statement.

The effect is the same as if the JCL statements of the cataloged procedure appeared in the input stream in the place of the EXEC statement that calls the procedure. If necessary, you can modify the cataloged procedure by a process known as overriding.

Before putting a procedure into the procedure library, you may want to test it. This can be done by converting the procedure to an in-stream procedure. An in-stream procedure is a set of JCL statements that can be used repeatedly by referencing it in an EXEC statement. After testing the procedure, keep it in card form and simply insert it in the input stream whenever you want to use it. Cataloged and in-stream procedures are not syntax checked until an EXEC statement that references the procedure is syntax checked. Therefore, be sure the job includes an EXEC statement that calls the procedure you want to test.

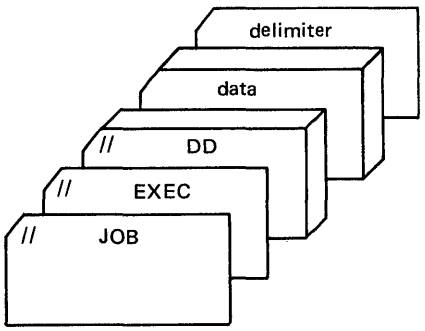
Another advantage of in-stream procedures is that they can give you the facility of a cataloged procedure without being placed in the procedure library.

Processing Your Job

To have a job processed, submit the JCL statements and any related input data to the operating system through an input/output (I/O) device chosen by the operator. The input unit can be a card reader, a magnetic tape, a terminal, or a direct access device. The sequence of JCL statements and input data for all the jobs being submitted through an input unit is called the input stream.

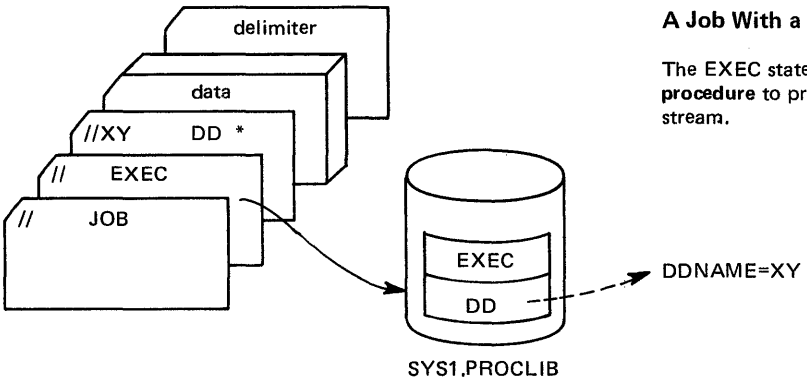
A job control language (JCL) statement consists of one or more 80-byte records. Many jobs are submitted to the operating system for execution in the form of 80-column punched cards. The operating system is able to distinguish a job control statement from data included in the input stream. In columns 1 and 2 of all the statements except the delimiter statement, code //. For the delimiter statement, code /*. For a comment statement, code //* in the first three columns.

A job can be simple or complicated; you can have a procedure in the input stream or call a cataloged procedure. Figure 5 shows some examples of what jobs can look like. Although only one example shows the use of the JES2 statements, these statements could have been placed with all of the jobs. (JES3 statements can also be used with any of the jobs and are placed after the JOB statement.)



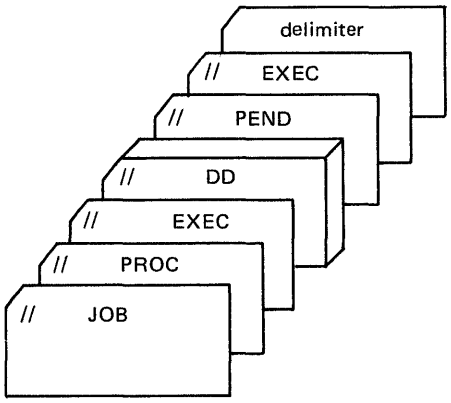
A Job With One Job Step

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



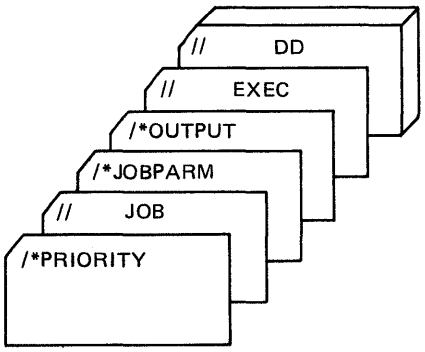
A Job With a Cataloged Procedure

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



A Job With an In-stream Procedure

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



A Job With JES2 Statements

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

Figure 5. A Job in the Input Stream

Figure 6 shows a job that contains several job steps: a compilation, a link-edit, and a program.

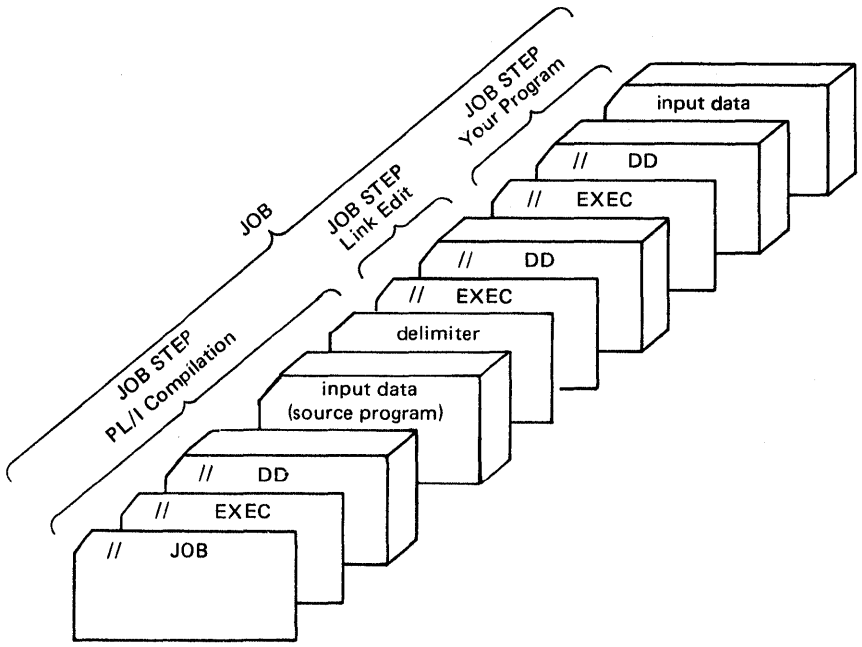


Figure 6. A Job with Several Job Steps

Figure 7 shows how several jobs run one after another through the input stream. Your job would be one job in a group of jobs that make up an input stream.

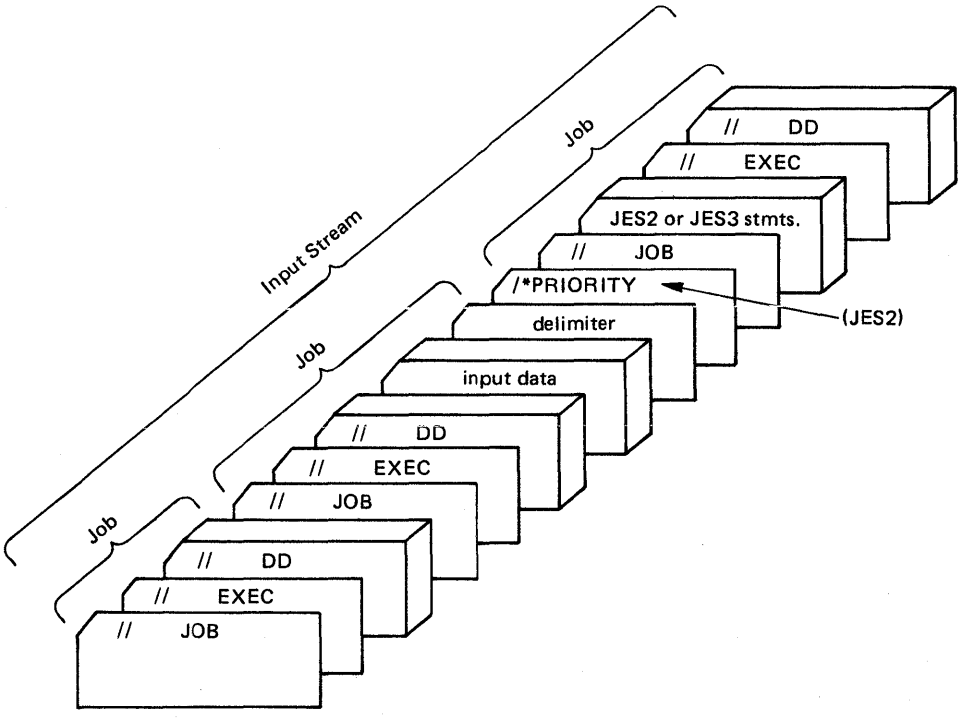


Figure 7. Job Boundaries in the Input Stream

To execute a program, define its requirements for resources and data. For example, if the job is to use only real storage, that is, it cannot be paged, you must indicate this on a JCL statement. You can also request certain units and volumes to be used and define the amount of space required by the job. If you request dynamic allocation, you are requesting resources during program execution as they are needed.

Define temporary or nontemporary data sets or request the use of a nontemporary data set that is cataloged. Whether a data set is old or new, should be kept or deleted, are a few of the options you can define in establishing the disposition of data sets in the job.

This section contains seven topics:

- Requesting Storage for Execution of a Program
- Requesting Units and Volumes
- Requesting Space for Non-VSAM Data Sets
- Mass Storage System (MSS) Considerations
- Dynamically Allocating and Deallocating Data Sets
- Identifying Data Sets to the System
- Disposition Processing of Non-VSAM Data Sets

Requesting Storage for Execution of a Program

In OS/VS, storage available for a program consists of real storage and virtual storage:

- **Real storage** is the storage of System/370 from which the central processing unit can directly obtain instructions and data and to which it can directly return results.
- **Virtual storage** is addressable space that appears to the user as real storage, from which instructions and data are mapped into real storage locations. The user address space is 16 million bytes which consists of the commonly addressable system storage, the nucleus, and the private address space (which includes the user's region).

When a program is selected, it is brought into virtual storage and divided into pages (a page is 4K in VS2). The supervisor is responsible for transferring pages of a program into real storage for execution. This paging is done automatically by the supervisor; to you, it appears as if the entire program exists in real storage. (The concept of paging is described in greater detail in the **Introduction to Virtual Storage in System/370.**)

When to Request Real Storage

For most programs, the supervisor transfers pages of a program to real storage as they are required for execution; not all pages of a program are necessarily in real storage at one time and the pages that are in real storage at once do not necessarily occupy contiguous space. Certain programs, however, must have all their pages in contiguous real storage while they are executing—they cannot be paged **during** execution. The programs include:

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

These programs must be placed into an area of virtual storage called the nonpageable dynamic area, whose virtual addresses are identical to real addresses; they are the only programs for which you should request **real** storage. If a job or job step must not be paged during execution, identify it by coding ADDRSPC=REAL on either the JOB or the EXEC statements. Request the amount of real storage needed with the REGION parameter.

Specifying Storage Requirements with the REGION Parameter

The meaning of the REGION parameter differs depending on whether the program can be paged during execution (if ADDRSPC=VIRT is coded or implied) or cannot be paged during execution (if ADDRSPC=REAL is coded).

When ADDRSPC=VIRT is coded or implied, two values are established internally from either the REGION parameter or an installation-supplied default. When ADDRSPC=REAL is coded, one value is established internally from either the REGION parameter or the installation-supplied default. These internal values are used to limit all GETMAINs. (For further information, see **OS/VS2 System Programming Library: Supervisor**, and the sections on the ADDRSPC and REGION parameters in this publication.)

The amount of space requested must include any additional requests the program makes during its execution (for example, a request made with the GETMAIN macro instruction). Also, the amount of storage requested must include sufficient space for the task termination function. Task termination invokes certain system resource managers that can issue GETMAIN macro instructions for space in the user's region. The region must have enough unallocated storage during task termination to allow the task termination function to complete.

When ADDRSPC=REAL is coded, the minimum region size must be 8K if the program to be executed is reenterable and resides in an authorized library, and 12K in all other cases. Note that this is the minimum region for successful execution, but not necessarily the minimum region size for successful job completion. It is suggested that programs to be run in an ADDRSPC=REAL environment perform as much clean-up as possible before terminating.

Example of Requesting Storage

The purpose of this job is to indicate how to request storage for a program when it is important that it not be paged.

```
//OBJ       JOB     BROWN,CLASS=D,MSGLEVEL=1
//STEP1     EXEC    PGM=REAL,REGION=20K,ADDRSPC=REAL
//DD1       DD      DSN=DISK1,DISP=OLD
//STEP2     EXEC    PGM=VIRT,REGION=75K,ADDRSPC=VIRT
//DD2       DD      DSN=DISK2,DISP=OLD
```

1. The JOB statement assigns jobs to class D and requests all JCL statements and messages to be printed.
2. STEP1 is to be executed in real storage.
3. STEP2 is to be executed in virtual storage.

Requesting Units and Volumes

On the DD statement defining a data set, indicate the device and volume on which the data set can be found or will be written by specifying unit and volume information. Input/output devices are grouped according to class; a device class is a kind of device: direct access, magnetic tape, unit record, graphic, and communications equipment. A **unit** is a particular device: a 2314 direct access device, a 1403 printer, etc.; a **volume** is a section of auxiliary storage that is serviced by a single read/write mechanism—for example, a reel of magnetic tape, a drum, or a disk pack.

Device status can affect the device eligibility for allocation. Figure 8 shows the various devices and the possible status each may have.

<table>
<tr><th rowspan="2">Status</th><th colspan="5">Device Type</th></tr>
<tr><th>Direct Access</th><th>Tape</th><th>Unit Record</th><th>Graphic</th><th>Teleprocessing</th></tr>
<tr><td>Online</td><td colspan="5">Eligible for allocation</td></tr>
<tr><td>Offline</td><td colspan="4">Eligible for allocation when the operator brings device online</td><td>Eligible for allocation</td></tr>
<tr><td>Pending Unload</td><td colspan="2">Eligible for allocation when volume is specifically requested</td><td colspan="3">not applicable</td></tr>
<tr><td>Pending Offline</td><td colspan="2">Eligible for allocation when the operator brings the device online and when the volume is specifically requested</td><td colspan="2">Eligible for allocation when the operator brings the device online</td><td>not applicable</td></tr>
</table>

Figure 8. How Device Status Affects Eligibility for Allocation

Specifying Volume Information

Data sets exist on direct access and magnetic tape volumes which must be mounted on devices before they can be used. To inform the system on which volume an existing data set can be found, make a specific volume request; to create a new data set, make a specific or nonspecific volume request. If you request multiple disk volumes to be mounted in JES3, they must all be either mountable or permanently resident; a mixture of both is not allowed.

Specific Volume Requests

A **specific volume request** informs the system of the volume serial number of the volume required. Make a specific volume request for an existing data set; make either a **specific** or **nonspecific** volume request when creating a data set.

A specific request occurs when:

- Specifying the serial numbers in the SER subparameter of the VOLUME parameter, that is, VOL=SER=(948762,945231).
- Referring the system to an earlier specific volume request to copy the volume serial numbers by coding the name of a passed or cataloged data set or a previous DD statement in the REF subparameter of the VOLUME parameter. To refer the system to a passed or cataloged data set, code VOL=REF=dsname. To refer to a DD statement in the same step, code VOL=REF=*.ddname; in a preceding step, VOL=REF=*.stepname.ddname; or in a procedure step that is in a procedure called by a preceding step, VOL=REF=*.stepname.procstepname.ddname. (If you refer to a multi-device type VSAM data set, only the volume serial number of the first device type listed in the catalog will be used.)
- Passing the data set from an earlier step or from the catalog. The system obtains the volume serial numbers from the passed data set information or from the catalog; you need not code the VOLUME parameter unless requesting a private volume, coding a volume sequence number, or requesting additional volumes. If a cataloged data set is cataloged in, or is to be cataloged or uncataloged from, a private catalog other than JOBCAT and STEPCAT, then the system automatically allocates that private catalog to the job step. (The private catalog must be on a permanently resident volume in JES3). If this allocation is not successful, then the job fails.

Nonspecific Volume Requests

Nonspecific volume requests can be made only for new data sets. When making a nonspecific volume request, do not specify volume serial numbers. You need not code the VOLUME parameter unless you are requesting a private volume or a volume count.

There are four types of nonspecific volume requests that can be made:

1. A private volume for a temporary data set.
2. A private volume for a nontemporary data set.
3. A nonprivate volume for a temporary data set.
4. A nonprivate volume for a nontemporary data set.

How the system satisfies these different types of requests is described below. Since the system satisfies the first two types of requests in the same way, these two requests are described together.

- When making a nonspecific volume request for a private **direct access or tape** volume, the system always requests the operator to mount a volume. The operator should mount a volume whose space is unused. This allows you to have control over all space on the volume. Once mounted, the volume is assigned the use attribute of private.
- When making a nonspecific volume request for a nonprivate **direct access** volume that is to contain a temporary data set, the system assigns a public or storage volume that is already mounted, or requests the operator to mount a removable volume. If a mounted volume is selected, its use attribute is not affected. If a removable volume is mounted, it is assigned the use attribute of public.
- When making a non-specific request for a non-private tape volume, data management (OPEN) will satisfy this request by using any available, physically mounted, tape volume. This could result in the loss of user data. However, if you use labels on your tape volumes and specify the types of labels in the LABEL parameter loss of user data can usually be prevented.
- When you specify labels in the LABEL parameter, data management (OPEN) checks the first record of the tape. There are various error conditions that can occur during verification of the first record. These error conditions are described in OS/VS Tape Labels.

  When making a nonspecific volume request for a nonprivate **tape** volume that is to contain a temporary data set, the system assigns a public volume that is already mounted, or it requests the operator to mount a tape volume. Once mounted, the volume is assigned the use attribute of public.
- When making a nonspecific volume request for a nonprivate **direct access** volume that is to contain a nontemporary data set, the system assigns a storage volume if one is mounted. Otherwise, the request is treated as a nonspecific volume request for a private volume.

  When making a nonspecific volume request for a nonprivate **tape** volume that is to contain a nontemporary data set, the request is treated as a nonspecific volume request for a private volume.

Note: If your nonspecific volume request requires more than one unit from a group that contains both single and dual density tape drives, the system assigns the devices so that the single density drive is the first one used. The default density is the density of the single density drive. The operator may be requested to mount the volumes in a different order than assigned by the system.

Using Private Volumes

A **private volume** is one that can be used exclusively unless a specific request is made for that volume. Code PRIVATE as the first subparameter in the VOLUME parameter with both specific and nonspecific volume requests. When making a specific volume request for a direct access volume, code PRIVATE if you want a private volume; tape volumes for which you make a specific volume request are automatically made private, so you need not code the PRIVATE subparameter.

A volume already made private cannot be allocated to satisfy other nonspecific volume requests. Therefore, if you request a private volume, you will be the only user using that volume, unless another job makes a specific volume request for that volume.

If PRIVATE is coded or implied, the operating system automatically requests that the operator demount the volume after its last use in the job step unless RETAIN is coded or the data set is passed. If you expect to use a data set in a subsequent step for which you requested a private volume, code RETAIN in the VOLUME parameter to ensure that the volume is not demounted at the end of the step. Even if you specify RETAIN or a disposition of PASS, the volume can still be unloaded by the operator or allocated and demounted by another step in the same job or by another job.

Sharing Volumes Between Data Sets

To conserve space and use fewer volumes, request that data sets be assigned the same volume. Data sets on the same volume have **volume affinity.**

You can request volume affinity either:

- Implicitly, through catalog references or by specifying the same volume serial numbers for the data sets in the SER subparameter of the VOLUME parameter.
- Explicitly, by using the REF subparameter of the VOLUME parameter to indicate that volumes identified in the catalog or on an earlier DD statement in the job are to be assigned to the data set being defined.

Volume affinity influences the allocation of devices. The system can modify a request for a specific number of units if a data set has volume affinity with at least one other data set. For examples of volume affinity, see "Example of UNIT and VOLUME Affinities" at the end of this section.

Multivolume Data Sets

If you are creating or extending a data set that can require more than one volume, request the maximum number of volumes that can be required in the **volume count** subparameter of the VOLUME parameter. The maximum number of volumes you can request is 255. For some jobs, each volume requested must be mounted on a unit before it can be used. For these jobs, request as many units as volumes. When making a specific volume request for more volumes than units, the system automatically indicates that the volumes on the same unit cannot be shared.

By coding the volume sequence number subparameter when reading or lengthening an existing multivolume data set, you can instruct the system to begin processing other than the first volume. Usually a volume sequence number is coded when you are defining an existing cataloged or passed data set.

Specifying Unit Information

Provide the system with the information it needs to assign a device to a data set in the UNIT parameter. To indicate what unit or type of unit you want, code one of the following:

- Unit address.
- Device type (generic name).
- User-assigned group name (esoteric name).

The **unit address** is a 3-character address made up of the channel, control unit, and unit number. For example, UNIT=180 indicates channel 1, control unit 8, and unit number 0. Specifying a unit address, however, limits unit assignment: the system can assign only that specific unit, and, if the unit is being used, the job must be delayed or canceled. Unit addresses should only be specified when necessary since these specifications restrict the system.

A **device type** corresponds to a particular set of features of input/output devices. When coding a device type, you allow the system to assign any available device of that device type. For example, UNIT=2314 indicates that you want the system to assign an available 2314 disk storage facility.

Each installation can also define **user-assigned group names** during system generation to signify a group of devices that may or may not all be of the same type. When coding a user-assigned group name, you allow the system to assign any available devices included in the group. For example, if the group named DISK includes all 2314 and 3330 disk storage facilities and you code UNIT=DISK, the system assigns an available 2314 or 3330 device. If the group named 2314A includes particular 2314's and you code UNIT=2314A, it could refer to one of

several 2314 devices. If the group consists of more than one device type, and more than one unit is requested, the units are allocated from the same device type. For example, if the group named TAPE includes both 3400-5 and 3400-6 devices, and you request two units by specifying UNIT=(TAPE,2), the system assigns either two 3400-5s or two 3400-6s. If there is an insufficient number of units of any single type to satisfy the request, the job is flushed.

If a group contains more than one device type or class (for example, SYSSQ can refer to all tape and direct access devices), you should not code the group name when defining an existing data set or requesting a specific volume. The volume on which the data set resides may require a device different from the one assigned to it. For example, if the data set resides on a tape volume, it must be assigned to a tape device.

The same is true if the data set resides on a 3348 Model 70F Data Module and the group name includes 3340 drives with and without the Fixed Head Feature. The 3348 Model 70F must be assigned to a 3340 with the feature. For more information on the Fixed Head Feature, see the **IBM 3340 Fixed Head Feature Users Guide.**

Only direct access devices can be simultaneously allocated for two or more jobs. Teleprocessing equipment is not allowed to be allocated more than once in the same job step. If a unit record, teleprocessing equipment, or graphics device is designated as a console, it is not eligible for allocation by a job.

Requesting More than One Unit

To increase operating efficiency, request multiple units for a multivolume data set or for a data set that may require additional volumes. When each required volume is mounted on a separate device, execution of the job step is not interrupted to allow the operator to demount and mount volumes. You should always request multiple units when the data set can be extended to a new volume if the data set resides on a permanently resident or reserved volume—permanently resident and reserved volumes cannot be demounted in order to mount a new volume.

You request multiple units by:

- Coding the unit count subparameter in the UNIT parameter.
- Requesting parallel mounting.

Request **parallel mounting** by coding P in place of the unit count subparameter when making a specific or non-specific volume request. The system counts the number of volumes requested (by counting the volume serial numbers specified on the DD statement or counting the volume serial numbers in cataloged or passed data sets). This is compared with the volume count, if it has been specified, and the system assigns the larger of the specified number of devices.

Deferred Mounting of Volumes

If the job step includes a data set that might not be used, depending on conditions determined in the job step, you can request (using the DEFER subparameter) that the system not mount the volume containing the data set until the data set is opened. This can save operator action of mounting volumes on direct access devices. Note: No other job step can use such a volume until the job step specifying DEFER ends. If DEFER is coded for a new data set which could be placed on a direct access device, then DEFER is ignored.

When You Do Not Have to Code the UNIT Parameter

The system can obtain unit information from sources other than the UNIT parameter. In these cases, you do not have to code the UNIT parameter:

- When the data set is cataloged. For cataloged data sets, the system obtains unit and volume information from the catalog. However, if VOL=SER=serial number is coded on a DD statement that defines a pre-existing data set, the system does not look in the catalog. In this case, you must code the UNIT parameter. Also, if you override the data set name on a procedure DD statement with the data set name of a cataloged data set, but do not override the UNIT parameter on that DD statement, the system will not search the catalog for unit information. Instead, the system will obtain unit information from the overridden DD statement. If you do override the data set name on a procedure DD statement and nullify the unit parameter on that DD statement, the system will search the catalog for unit information.
- When the data set is passed from a previous job step. For passed data sets, the system obtains unit and volume information from passed data set information. However, if VOL=SER=serial number is coded on a DD statement that defines a pre-existing data set, the system does not look in the passed data set information. In this case, you must code the UNIT parameter.
- When the data set is to use the same volumes assigned to an earlier data set, that is, VOLUME=REF=reference is coded. In this case, the system obtains unit and volume information from an earlier DD statement that specified the volume serial number or from the catalog.

In all of the cases listed above, code the UNIT parameter when you want additional devices assigned or when you want to influence device allocation. If the coded UNIT parameter is a subset of the unit type referenced, then it will be used. Otherwise, it is ignored. However, when JES3 looks in the catalog, it cannot determine whether or not a given device type is a subset of another device type. Errors might result if you request one device to be mounted on a conflicting device type (for example, a 3330 mounted on a 2314). Do not code the UNIT parameter when defining a data set included in the input stream. If UNIT is coded on a DD * or DD DATA statement, the job abnormally terminates.

Sharing a Unit Between Data Sets on Different Volumes

To conserve the number of devices used in a job step, you can request that an existing data set be assigned to the same device or devices as assigned to a data set defined earlier in the job step. When two or more volumes are assigned the same device, the volumes are said to have **unit affinity**. Unit affinity implies deferred mounting for all except one of the volumes, since all volumes cannot be mounted on the same device at the same time.

Request explicit unit affinity by coding UNIT=AFF=ddname on a DD statement. The ddname is the name of an earlier DD statement in the same job step. The data set defined on the DD statement that requests unit affinity is assigned the same device or devices as the data set defined on the named DD statement and must reside on the same device type. If the ddname refers to a DD statement that defines a dummy data set, the data set defined on the DD statement requesting unit affinity is assigned a dummy status. Unit affinity also exists on one DD statement when there are more volumes than units. This is implied unit affinity. See examples of unit affinity.

If all of the following conditions are present, the data set defined on the DD statement requesting unit affinity might be written over by the named data set:

- The named DD statement requests a scratch tape.

- The data set defined on the DD statement requesting unit affinity is opened prior to that on the named DD statement.
- The tape is not unloaded prior to the OPEN of the data set defined on the named DD statement and tape label positioning is not specified using the LABEL parameter. (Note that a tape unit that is allocated to more than one request is not unloaded (1) as a result of dynamic unallocation, or (2) when it is closed if FREE=CLOSE is specified.)

Note: Unit affinity cannot be requested for a new data set if the referenced request is for a direct access device.

Unit and Volume Affinities: Unit and volume affinity can occur in the same step and, within the step, on the same DD statement. Unit affinity is requested when UNIT=AFF=ddname is specified or when more volumes than units are requested on the same DD statement. Volume affinity is requested when two or more DD statements reference the same volume. There are three relationships possible between unit and volume affinity.

1. All volume affinity requests are unrelated to any of the unit affinity requests. For example,

```
//DD1    DD      VOL=SER=A
//DD2    DD      UNIT=AFF=DD1,VOL=SER=B
//DD3    DD      VOL=SER=(C,D)
//DD4    DD      VOL=SER=C
```

   Unit affinity is requested between DD1 and DD2. Volume affinity is requested between DD3 and DD4. Therefore, volume affinity requests are not related to unit affinity requests.

2. All volume affinity requests are contained in the unit affinity requests. For example,

```
//DD1    DD      VOL=SER=(A,D)
//DD2    DD      UNIT=AFF=DD1,VOL=SER=(A,B)
//DD3    DD      VOL=SER=X
```

   Unit affinity is requested between DD1 and DD2. Volume affinity is also requested between DD1 and DD2. Therefore, volume affinity requests are contained within the chain of unit affinity requests.

3. Some volume affinity requests are contained in the unit affinity requests, but not all. For example,

```
//DD1    DD      VOL=SER=A
//DD2    DD      UNIT=AFF=DD1,VOL=SER=B
//DD3    DD      VOL=SER=B
```

   Unit affinity is requested between DD1 and DD2. Volume affinity is requested between DD2 and DD3. Therefore, some volume affinity requests are contained in the chain of unit affinity requests, but not all.

If both unit and volume affinity do exist in the same step, sometimes only one requested affinity can be honored at a time. Figure 9 indicates what will happen when you code unit and volume affinity for either tape or direct access devices.

Relationship of unit and volume affinity requests	Tape	Direct Access
Unit and volume affinity requests unrelated.	Because there is no conflict, both unit and volume affinity requests are honored.	Because there is no conflict, both unit and volume affinity requests are honored.
All volume affinity requests contained in unit affinity requests.	All volumes will use the same unit; that is, volume affinity is ignored and unit affinity is honored.	For those volumes having volume affinity that are contained in the unit affinity requests, unit affinity is ignored. That is, they will share the same unit while the remaining requests in the unit affinity will use a different unit.
Some volume affinity requests contained in unit affinity requests.	For those volumes having volume affinity that are contained in the unit affinity requests, unit affinity is ignored. That is, they will share the same unit while the remaining requests in the unit affinity will use a different unit.	For those volumes having volume affinity that are contained in the unit affinity requests, unit affinity is ignored. That is, they will share the same unit while the remaining requests in the unit affinity will use a different unit.

Figure 9. Unit and Volume Affinity

Note: If a requested volume is mounted on an eligible permanently resident or reserved unit, it must be allocated to that unit regardless of any relationships to other requests. This is done because no dismount of that particular volume can take place.

Example of UNIT and VOLUME Affinities: The purpose of this job is to show several job steps that use either unit or volume affinity for their processing.

```
//AFFIN     JOB     (8526,831),WOON,CLASS=J,PERFORM=50
//STEP1     EXEC    PGM=TESTAFF
//DD1       DD      UNIT=2400,VOL=SER=111111
//DD2       DD      UNIT=AFF=DD1,VOL=SER=222222
//STEP2     EXEC    PGM=TESTAFF
//DD11      DD      UNIT=(3330,2),VOL=SER=(A,B)
//DD12      DD      UNIT=AFF=DD11,VOL=SER=(C,D)
//STEP3     EXEC    PGM=TESTAFF
//DD21      DD      UNIT=(3330,2),VOL=SER=(A,B)
//DD22      DD      UNIT=AFF=DD21,VOL=SER=(C,D)
//DD23      DD      UNIT=3330,VOL=SER=B
//STEP4     EXEC    PGM=TESTAFF
//DD31      DD      UNIT=(3330,2),VOL=SER=(E,F)
//DD32      DD      UNIT=AFF=DD31,VOL=SER=(G,H)
//STEP5     EXEC    PGM=TESTAFF
//DD41      DD      UNIT=2400,VOL=SER=(111111,222222)
//DD42      DD      UNIT=AFF=DD41,VOL=SER=(222222)
//STEP6     EXEC    PGM=TESTAFF
//DD51      DD      UNIT=3330,VOL=SER=(ABCDEF,GHIJKL)
//DD52      DD      UNIT=AFF=DD51,VOL=SER=(ABCDEF)
//STEP7     EXEC    PGM=TESTAFF
//DD61      DD      UNIT=2400,VOL=SER=111111
//DD62      DD      UNIT=2400,VOL=SER=111111
//DD63      DD      UNIT=AFF=DD61,VOL=SER=222222
```

1. The JOB statement assigns jobs to class J in performance group 50.
2. STEP1 assigns one unit for both volumes. Volume 111111 will be mounted first, then 222222 will be mounted when DD2 is opened. (This processing is true for both tape and direct access.)
3. STEP2 allocates two units to DD11 and volumes A and B are mounted. DD12 gets allocated to the same two units but volumes C and D will be mounted when DD12 is opened.
4. STEP3 is a direct access example of volume affinity for volume B. The actual allocation of units will cause volumes A and C to share one unit and volumes B and D to have their own units.
5. STEP4 is a direct access example. Assume that volume E is currently mounted and has been assigned the permanently resident or reserved attribute. In this case, since volume E cannot be dismounted, a separate unit will be allocated for it. Volume G will have its own unit and volumes F and H will share one unit. Therefore, three volumes will be allocated for these requests, instead of two, because of the permanently resident or reserved mount attributes.
6. STEP5 is a tape example. Volume affinity is ignored between the DD statements because only one tape data set for each tape volume can be open at a time.
7. STEP6 is a direct access example where unit affinity is ignored for the common volume. Volume ABCDEF of both DD statements will share the same unit while the remaining request (GHIJKL) will use a different unit.
8. In STEP7, unit affinity is requested between DD61 and DD63. Volume affinity is requested between DD61 and DD62. Because there is a volume affinity request (DD62) that is not contained in the chain of unit affinity requests, the UNIT=AFF=DD61 specification is ignored for DD63. STEP7 allocates two units; one for volume 111111, and another for volume 222222.

For more information, see **OS/VS2 System Programming Library: Job Management.**

Example of Requesting Units and Volumes

This job shows the unit and volume parameters.

```
//TEST          JOB     WIBORG,CLASS=C
//STEP1         EXEC    PGM=TESTSYSO
//DD11          DD      DSN=A01DD1,UNIT=3330,DISP=(,PASS),
//                      SPACE=(TRK,1),VOL=SER=333001
//STEP2         EXEC    PGM=TESTSYSO
//DD21          DD      DSN=SYSLIB,UNIT=2314,VOL=(PRIVATE,SER=123456),
//                      DISP=OLD
//DD22          DD      DSN=SYSABC,UNIT=AFF=DD21,VOL=SER=777777,
//                      DISP=(OLD,KEEP)
//DD23          DD      DSN=SYSTAPE,UNIT=(2400,P,DEFER),DISP=OLD,
//                      VOL=SER=(240001,240002,240003,240004,240005)
//DD24          DD      DSN=SYSDISK,DISP=(SHR,KEEP),UNIT=(,P),
//                      VOL=SER=(333005,333008,333010)
//DD25          DD      UNIT=2314,VOL=REF=*.DD21,SPACE=(TRK,(5,2))
//DD26          DD      UNIT=3330,VOL=REF=SYSDISK,SPACE=(TRK,(10,5))
```

1. The job is assigned to class C.
2. DD11 defines a new data set named A01DD1. It is to be on volume 333001 which is mounted on a 3330 device.
3. DD21 defines an old data set named SYSLIB that exists on a private volume, 123456. The volume is mounted on a 2314 device.
4. DD22 defines an old data set named SYSABC that is to be kept after this job step is complete. SYSABC is on volume 777777. This volume is to be mounted on the same 2314 device as the volume defined on DD21.
5. DD23 defines an old data set named SYSTAPE. There are five volumes that are to be mounted only after the data set is opened (caused by the DEFER subparameter). The P requests parallel mounting; that is, all five volumes are to be mounted at the same time on five different 2400 devices.
6. DD24 defines an old data named SYSDISK that can be shared by another job since it will only be read. It is to be kept after this job step. The number of units used is determined by the number of volumes requested.
7. DD25 is a temporary data set (no DSNAME specified) and therefore, assumes a disposition of NEW,DELETE. The volume to be used is the same one used in STEP2 DD21; that is, volume 123456.
8. DD26 is also a temporary data set. The backward reference for volume information is to STEP2 DD24 where the data set named SYSDISK is located.

Requesting Space for Non-VSAM Data Sets

You must request space for every non-VSAM data set created on a direct access volume. To request space, code the SPACE parameter on the DD statement that defines the data set. The SPACE parameter provides two ways to request space:

- Tell the system how much space you want and let the system assign specific tracks.
- Tell the system the specific tracks on which you want the data set written.

Letting the system assign specific tracks is the easiest and most frequently used method of requesting space. Only the unit of measurement to be used to compute the space requirement and how many of the units of measurement the data set requires needs to be specified. In addition, this form of the SPACE parameter offers several options:

- A secondary quantity, to be used if the data set runs out of space.
- Space for a directory or index.
- Release of unused space.
- Contiguous space.
- Whole cylinders.

OS/MVT and VS2 Release 1 (SVS) included the SPLIT and SUBALLOC parameters for requesting space for a group of data sets on a single direct access volume. These two parameters are now internally converted to SPACE requests. SUBALLOC requests are not eligible for virtual input/output (VIO).

The Basic Request: Unit of Measurement and Primary Quantity

To have the system assign specific tracks, specify only the **unit of measurement** the system should use to allocate space and the **primary quantity** of space needed. As the unit of measurement, you can specify:

- Average block length of the data, for blocks.
- TRK, for tracks.
- CYL, for cylinders.

As the primary quantity, code an integer, indicating how many blocks, tracks, or cylinders are required.

It is easiest to specify an average block length: the system will allocate the least number of tracks required to contain the number of blocks specified. Specifying block length also maintains device independence; you can change the device type in the UNIT parameter without altering the space request or code a group name that includes different direct access devices in the UNIT parameter.

When specifying TRK or CYL, compute the number of tracks or cylinders required; consider such variables as the device type, track capacity, tracks per cylinder, cylinders per volume, data length (blocksize), key length, and device overhead. These variables, and examples of estimating space requirements for partitioned and indexed sequential data sets, are described in **OS/VS2 MVS Data Management Services Guide.**

Cylinder allocation allows faster input/output of sequential data sets than does track allocation. When requesting space in terms of average block length, the system will allocate tracks to contain the request unless you code ROUND as the last subparameter in the SPACE parameter. The system will then allocate the smallest number of cylinders needed to contain the request.

How the System Satisfies Your Primary and Secondary Request

Enough available space must exist on one volume to satisfy the primary request. If enough space is not available on a single volume, the system will terminate the job or search another volume, depending on the type of volume request made:

Specific volume request (for example, volume serial numbers are specified): If sufficient space is not available on the first volume specified, the job is terminated.

Nonspecific volume request (for example, the system chooses the volume): If space is not available on the first volume chosen, the system will choose another volume and continue the search, causing volumes to be mounted if necessary, until a volume with sufficient space is found or the operator cancels the job.

Note: If a non specific volume request for an ISAM data set is made via a single DD statement and space is not available on the volume selected for allocation, the job fails.

The system attempts to allocate the primary and secondary quantity in contiguous tracks or cylinders. If contiguous space is not available, the system satisfies the request with up to five noncontiguous **extents** (blocks) of space. (If user labels are specified — that is, you code SUL in the LABEL parameter—the system allocates up to four noncontiguous extents of space. The system allocates a track for user labels separate from the primary quantity; this one track is considered an extent, and therefore, up to four additional extents can be allocated to satisfy the primary quantity.)

A Secondary Request for Space

In the primary quantity, you need not anticipate all future demands for space for a data set. Code a **secondary** request for space to be used only if the data set exceeds its allocated space. Do this by coding an integer following the primary quantity that indicates how much additional space should be allocated. For data sets whose disposition is NEW or MOD, this space is allocated on the same volume as the primary quantity until: (1) there is not enough space available on the volume to allocate the secondary quantity, or (2) a total of 16 extents, less the number of extents for primary quantity and user label space, have been allocated to the data set. (BDAM data sets cannot be extended.) If either of these conditions is satisfied, the system must allocate the secondary quantity on another volume. However, this can be done only if you request more than one volume in the VOLUME parameter (for a nonspecific volume request, code PRIVATE; for a specific volume request, request more volumes than devices).

When allocating a secondary quantity for a data set whose disposition is OLD (in other words, a data set that is preallocated or is being written over), the system will go to the next volume, if one is specified, and see if there is already a secondary quantity allocated there. If you did specify another volume and there is already a secondary quantity, the system will use that space instead of making another allocation or will allocate space if no space is already allocated there for the data set. If you didn't specify another volume, the space will be allocated on the current volume.

A secondary quantity can be requested when creating a data set or when retrieving an existing data set, whether or not you coded a secondary quantity in the original request. A secondary request for an existing data set is in effect only for the duration of the job step and overrides an original request if one was made.

If you specify SPACE in terms of average block length, code the maximum block length of the data in either the DCB macro instruction or the BLKSIZE subparameter of the DCB parameter on the DD statement: the system uses the maximum block length to compute how many additional tracks to allocate.

Requesting Directory Space for a Partitioned Data Set

To create a partitioned data set, request a primary quantity large enough to include space for a directory. A directory is an index used by the system to locate members in a partitioned data set. It consists of 256-byte records, and you must specify, as the third quantity in the SPACE parameter, how many records the directory is to contain. The directory is included in the beginning of the primary space, which must be large enough to contain the directory. Request enough directory space to allow for growth of the data set: you cannot lengthen the directory as you can lengthen the data set itself by requesting a secondary quantity. If the directory runs out of space, recreate the data set. For a complete description of the directory, including details on member entries that will enable you to compute how many records to request, see **OS/VS2 MVS Data Management Services Guide.**

Requesting Index Space for an Indexed Sequential Data Set

If you are creating an indexed sequential data set that occupies more than one cylinder, and are not defining the index on a separate DD statement, you can request index space in addition to a primary quantity. (Request index space as the third quantity in the SPACE parameter. The space request for an indexed sequential data set must be in terms of cylinders or absolute track allocation.) The system determines whether the request is for a directory or an index by examining the DSORG subparameter of the DCB parameter on the DD statement. DCB=DSORG=IS or DCB=DSORG=ISU must be included on any DD statement defining an indexed sequential data set.

The index quantity is added to the primary quantity when considering the space requirements.

Assigning Specific Tracks

You can request that specified tracks on a volume be allocated to a data set. This is the most stringent request for space: if any of the tracks requested are occupied, the space cannot be allocated and the job is terminated. An example of where specific track allocation is required is a data set that is to reside under the fixed heads of a 3348 Model 70F Data Module (cylinders 1-5).

To request specific tracks, you must code:

- ABSTR as the first subparameter, indicating absolute tracks.
- A primary quantity, specifying the number of tracks to be allocated.
- The relative track number of the first track to be allocated.

For a partitioned data set, specify how many records you want allocated for a directory. If requesting a user-label track, this track will be the first of the space requested.

If defining an indexed sequential data set using absolute track allocation, the number of tracks for the index, primary, or overflow areas must be equal to an integral number of cylinders and on a cylinder boundary. All of the DD statements defining the indexed sequential data sets must request specific tracks.

Example of Requesting Space

One purpose of this job is to request space for two temporary data sets. The following steps refer to these data sets for volume information.

```
//ALLOC      JOB      (3416,354),STONER,MSGLEVEL=1,MSGCLASS=C
//STEP1      EXEC     PGM=TESTSYS0
//DD11       DD       UNIT=2314,DISP=(,PASS),SPACE=(TRK,(10,5))
//DD12       DD       UNIT=3330,DISP=(,PASS),SPACE=(TRK,(10,5))
//SYSABEND   DD       SYSOUT=L
//STEP2      EXEC     PGM=TESTSYS0
//DD1        DD       DSN=*.STEP1.DD11,DISP=(OLD,DELETE,DELETE)
//DD2        DD       VOL=REF=*.STEP1.DD12,SPACE=(TRK,(3,1)),UNIT=3330
//SYSABEND   DD       SYSOUT=L
```

1. The JOB statement specifies that all job related output is to be printed and that system messages for the job are to be written to output class C.
2. STEP1 defines two temporary data sets. Step 2 refers to these data sets for volume information.
3. The space requirements for these requests indicate that for DD11 and DD12 in STEP1 you want 10 primary and 5 secondary tracks; and for DD2 in STEP2 you want 3 primary and 1 secondary track.

Mass Storage System (MSS) Considerations

Mass storage volumes are accessed on virtual direct access devices. All previously defined descriptions of direct access device resource requests apply, with several additional functions also available. The mass storage volume device type is 3330V.

Mass Storage Volume Groups

The mass storage system (3850) can contain up to 4,720 mass storage volumes (3330V). To assist the installation in managing the volumes, the mass storage system utilities are used to assign the volumes to groups. When creating a new data set with a nonspecific request, the desired group can be specified using MSVGP=id. The system then selects the best volume for the requirements from the specified group.

The installation can define as many groups as necessary; one group and its name are standard in all systems (SYSGROUP). The installation then assigns each mass storage volume to a user group, SYSGROUP, or to no group.

Nonspecific Volume Requests for Mass Storage Volumes

Previously defined descriptions of nonspecific DASD volume requests apply to mass storage volumes. The type of request can be modified by the MSVGP parameter that specifies an installation defined subset of all mass storage volumes to be used by the system to satisfy the request. MSVGP implies a private volume. The system will select a volume from the defined group that has sufficient space to satisfy the space requirements of the DD statement. (See the section on mass storage volume control in **OS/VS Mass Storage System (MSS) Services: General Information** for the selection of MSVGP volumes to satisfy space requirements.) If you code the MSVGP parameter, the VOLUME parameter can be used to specify a volume count, but must not be used for volume serial numbers. VOLUME=PRIVATE is redundant when MSVGP is used.

If MSS Enhancements (SU24) is installed on your system, you can specify that data sets be allocated to different volumes by coding the ddname operand on the MSVGP parameter. It may be desirable to specify different volumes for two data sets, for example, when an existing data set containing a critical master file is used for input and a new data set is created for the output master file.

If MSVGP is not specified—

- when you make a nonspecific request for a private mass storage volume, the system always causes a default group of volumes to be used (MSVGP=SYSGROUP).
- when you make a nonspecific request for a non-private mass storage volume that is to contain a temporary data set, the system assigns a public or storage mass storage volume that is already mounted if one is available. Otherwise, the request is treated as a nonspecific volume request for a private volume.
- when you make a nonspecific request for a non-private mass storage volume that is to contain a nontemporary data set, the system assigns a storage mass storage volume, if one is mounted. Otherwise, the request is treated as a nonspecific volume request for a private volume.

Specific Volume Requests for Mass Storage Volumes

Previously defined descriptions of specific DASD volume requests (direct access storage volumes) also apply to mass storage volumes.

Because there is no operator involvement or decision making in mounting mass storage volumes, it is recommended (for data integrity purposes) that all permanent data sets on mass storage volumes be cataloged. All specific requests for these data sets should always reference the volumes using the catalog, not the VOLUME parameter. Reference to the catalog is required when extending an existing multivolume data set to one or more volumes. The reason is that the system must know all volumes on which the data set currently resides before it selects the new volume. Parallel mounting must also be specified to ensure proper multivolume extensions.

Requesting Space for Non-VSAM Data Sets on Mass Storage Volumes

When an installation defines mass storage volume groups, each group is given a default for space. Specific volume requests for new data sets require the SPACE parameter. Nonspecific volume requests with the MSVGP parameter can optionally specify the SPACE parameter. Nonspecific volume requests without the MSVGP parameter can optionally specify the SPACE parameter if the request will default to MSVGP=SYSGROUP. If other types of space attributes are desired, the SPACE parameter can be coded to override the specified default. Neither directory nor index quantities can be provided in the default; therefore, the SPACE parameter must be coded for new BPAM or ISAM data sets on mass storage volumes.

Before using mass storage volumes, refer to **OS/VS Mass Storage System (MSS) Services: General Information and OS/VS Mass Storage System (MSS) Services: Reference Information.**

Dynamically Allocating and Deallocating Data Sets

Dynamic allocation allows you to acquire resources as they are needed. One reason to use dynamic allocation is that you may not know all of the device requirements for a job prior to execution. Another reason is that it allows resources to be used more efficiently; that is, resources can be acquired just before their use and/or released immediately after use. (Resources, as used here, refer to a ddname-data set combination with its associated volumes and devices, if any.) The number of dynamic allocations indicated by coding the DYNAM and DYNAMNBR parameters are used to establish a control limit for tracking resources held in anticipation of reuse.

You can dynamically deallocate resources during the execution of a job step (at the time the data set is closed) by coding the FREE=CLOSE parameter. If you do dynamically deallocate a resource at close time, it cannot be reopened in the same step. If you do not want to dynamically deallocate the resource, either specify nothing or specify FREE=END to let the system deallocate the resources at the end of the job step.

For more information on how to use dynamic allocation and deallocation and the control limit, see **OS/VS2 System Programming Library: Job Management.**

Example of Dynamically Deallocating Data Sets

```
//PROS    JOB     CLASS=A,MSGLEVEL=(2,0),PERFORM=70
//STEP1   EXEC    PGM=TEST,DYNAMNBR=4,PARM=(P3,123,MT5)
//DD1     DD      DYNAM
//DD2     DD      DYNAM
//OUT1    DD      SYSOUT=C,FREE=CLOSE
//OUT2    DD      SYSOUT=A
//SYSIN   DD      *
          .
          .
          .
          data
          .
          .
          .
/*
```

1. The JOB statement specifies that this job will be processed in class A in performance group 70. Only JCL statements will be printed.
2. The control limit is the sum of the number of DD statements coded and the value coded in the DYNAMNBR parameter; in this case, 9. If this control limit is exceeded and a request for another dynamic allocation is made, the request is not honored unless resources can be deallocated so that the control value is not exceeded.
3. When OUT1 is closed, it is immediately ready for printing.

Identifying Data Sets to the System

Specifying the DDNAME Parameter

The DDNAME parameter is most often used in cataloged procedures and in job steps that call procedures. It is used in cataloged procedures to postpone defining data in the input stream until a job step calls the procedure. (Procedures cannot contain DD statements that define data in the input stream; that is, DD * or DD DATA statements). It is used in job steps that call procedures to postpone defining data in the input stream on an overriding DD statement until the last overriding DD statement for a procedure step. (Overriding DD statements must appear in the same order as the corresponding DD statements in the procedure).

When You Code the DDNAME Parameter

When the system encounters a DD statement that contains the DDNAME parameter, it saves the ddname of that statement. The system also temporarily saves the name specified in the DDNAME parameter so that it can relate that name to the ddname of a later DD statement. Once a DD statement with that corresponding name is encountered, the name is no longer saved. For example, if the system encounters this statement

```
//XYZ        DD        DDNAME=PHOB
```

the system saves XYZ and, temporarily, PHOB. Until the ddname is encountered in the input stream, the data set is a dummy data set.

When the system encounters a statement whose ddname has been temporarily saved, it does two things. It uses the information contained on this statement to define the data set; it associates this information with the name of the statement that contained the DDNAME parameter. The value that appeared in the DDNAME parameter is no longer saved by the system. To continue the above example, if the system encounters this statement

```
//PHOB       DD        DSNAME=NIN,DISP=(NEW,KEEP),UNIT=2400
```

the system uses the data set name and the disposition and unit information to define the data set; it also associates the ddname of the statement that contained the DDNAME parameter with this information. In this example, the ddname used is XYZ; the ddname PHOB is no longer saved. The data set is now defined, just as it would be if you had coded

```
//XYZ        DD        DSNAME=NIN,DISP=(NEW,KEEP),UNIT=2400
```

The system associates the ddname of the statement that contains the DDNAME parameter with the data set definition information. It does not use the ddname of the later statement that defines the data set. Therefore, any references to the data set, before or after the data set is defined, must refer to the DD statement that contains the DDNAME parameter, not the DD statement that defines the data set. The following sequence of control statements illustrates this:

```
//DD1        DD        DDNAME=LATER
                .
                .
                .
//LATER      DD        DSN=SET12,DISP=(NEW,KEEP),UNIT=2314,
//                     VOLUME=SER=46231,SPACE=(TRK,(20,5))
                .
                .
                .
//DD12       DD        DSN=SET13,DISP=(NEW,KEEP),VOLUME=REF=*.DD1,
//                     SPACE=(TRK,(40,5))
```

When you want to concatenate data sets, the unnamed DD statements must follow the DD statement that contains the DDNAME parameter, not the DD statement that defines the data set. The following sequence of control statements illustrates this:

```
//DDA       DD      DDNAME=DEFINE
//          DD      DSN=A.B.C,DISP=OLD
//          DD      DSN=SEVC,DISP=OLD,UNIT=2314,VOL=SER=52226
            .
            .
            .
//DEFINE    DD      *
            data
/*
```

You can use the DDNAME parameter up to five times in a job step or procedure step. However, each time the DDNAME parameter is coded, it must refer to a different ddname.

Specifying the DSNAME Parameter

When creating a data set, use the DSNAME parameter to assign a name to the data set. The data set name is part of the information stored with the data set on a volume. Later, when another job step or job wants to use the data set, it identifies the data set name in the DSNAME parameter; the system uses the data set name to locate the data set on the volume.

How you code the DSNAME parameter depends on the type of data set and whether the data set is nontemporary or temporary.

Creating or Retrieving a Nontemporary Data Set

If the data set is nontemporary, you can identify:

- A permanent data set by coding DSNAME=dsname.
- A member of a nontemporary partitioned data set by coding DSNAME=dsname(member name).
- A generation of a nontemporary generation data group by coding DSNAME=dsname(number).
- An area of a nontemporary indexed sequential data set by coding DSNAME=dsname(area name).

Nontemporary Data Sets

When a nontemporary data set is created, it is assigned a name in the DSNAME parameter and is assigned a disposition of KEEP or CATLG. (A data set assigned a disposition of KEEP may be assigned a disposition of CATLG by a later job step or job). The name assigned to a nontemporary data set must be specified in the DSNAME parameter by all other steps and jobs that want to use the data set.

A nontemporary data set name can be either an unqualified or qualified name. An unqualified data set name consists of 1 through 8 characters. The first character must be an alphabetic or national (@,#,$) character; the remaining characters can be any alphameric or national characters, a hyphen, or plus zero (0-12 punch).

A qualified data set name consists of 1 through 44 characters (including periods), except when the qualified name identifies a generation data group. In this case, the data set name may consist of only 1 through 35 characters (including periods). For each eight characters or less there must be a period, and the first character of the name and the character following a period must be an alphabetic or national (@,#,$) character.

When requesting a data set that is cataloged on a control volume or a private catalog, the system attempts to mount this control volume if it is not already mounted. After the system obtains the pointer to the requested data set, the control volume or private catalog can then be demounted by the system if the unit on which it was mounted is required by another volume. The control volume or private catalog is assigned to the job step and is available for disposition processing when the job step ends.

In the following cases, the control volume or private catalog is not mounted when disposition is processed:

- The job fails or abnormally terminates and data sets with a conditional disposition of CATLG or UNCATLG have been passed but not received.
- A job step is deallocated during system warmstart.

Members of a Partitioned Data Set

A partitioned data set consists of independent groups of sequential records, each identified by a member name in a directory. When you want to add a member to a partitioned data set or retrieve a member, specify the partitioned data set name and follow it with the member name. The member name is enclosed in parentheses and consists of 1 to 8 characters. The first character must be an alphabetic or national (@,$,#) character, the remaining characters can be any alphameric or national characters.

Generations of a Generation Data Group

A generation data group is a collection of chronologically related data sets that can be referred to by the same data set name. When you want to add a generation to a generation data group or retrieve a generation, specify the generation data group name and follow it with the generation number. The generation number is enclosed in parentheses and the number is a zero or a signed integer. A zero represents the most current generation of the group, a negative integer (for example, -1) represents an older generation; a positive integer (for example, +1) represents a new generation that has not as yet been cataloged.

To retrieve all generations of a generation data group (up to 255 generations), code only the group name in the DSNAME parameter and the DISP parameter.

A complete discussion of creating and retrieving generation data sets is contained in "Creating and Retrieving Generation Data Sets."

Areas of an Indexed Sequential Data Set

The areas used for an indexed sequential data set are the index, prime, and overflow areas. When you are creating the data set and define any of these areas on a DD statement, you must identify the data set name and follow it with the area name you are defining. The area name is enclosed in parentheses and is either PRIME, INDEX, or OVFLOW. If you are using only one DD statement to define the entire data set, code DSNAME=dsname or DSNAME=dsname(PRIME). When you retrieve the data set, you code only the data set name; you do not include the terms PRIME, INDEX, or OVFLOW.

Creating or Retrieving a Temporary Data Set

If the data set is temporary, you can identify:

- A temporary data set by coding DSNAME= & & dsname.
- A member of a temporary partitioned data set by coding DSNAME= & & dsname(member name).
- An area of a temporary indexed sequential data set by coding DSNAME= & & dsname(area name).

Temporary Data Sets

Any data set that is created and deleted within the same job is a temporary data set. A DD statement that defines a temporary data set need not include the DSNAME parameter; the system generates one for you.

If you do include the DSNAME parameter, the temporary data set name can consist of 1 through 8 characters and is preceded by two ampersands (& &). The character following the ampersands must be alphabetic or national (@,#,$) characters; the remaining characters can be any alphameric or national characters. (A temporary data set name that is preceded by only one ampersand is treated as a temporary data set name as long as no value is assigned to it either on the EXEC statement for this job step when it calls a procedure, or on a PROC statement within the procedure. If a value is assigned to it by one of these means, it is treated as a symbolic parameter).

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

If you attempt to keep or catalog a temporary data set (you specify a disposition of KEEP or CATLG in the DISP parameter), the system changes the disposition to PASS and the data set is deleted at job termination. However, this change is not made for a data set on a tape volume when the following conditions exist:

- The data set is new.
- The data set is not assigned a name.
- You specify a status of OLD or SHR in the DISP parameter.
- You specify DEFER in the UNIT parameter.

The data set is deleted at job termination, but the system tells the operator to keep the volume on which the data set resided during the job. If you code a conditional disposition for temporary data sets, it is ignored.

To simplify processing of temporary data sets, see "Using Virtual Input/Output (VIO) for Temporary Data Sets".

Members of a Temporary Partitioned Data Set

When adding a member to a temporary partitioned data set or retrieving a member during the job, specify the partitioned data set's temporary name and follow it with the member name. The member name is enclosed in parentheses and consists of 1 through 8 characters. The first character must be an alphabetic or national (@,$,#) character; the remaining characters can be any alphameric or national characters.

Areas of a Temporary Indexed Sequential Data Set

The areas used for indexed sequential data set are the index, prime, and overflow areas. When you are creating a temporary indexed sequential data set and define any of these areas on a DD statement, you must identify the data set's temporary name and follow it with the area name you are defining. The area name is enclosed in parentheses and is either PRIME, INDEX, or OVFLOW. If you are using only one DD statement to define the entire temporary data set, code DSNAME= & & dsname or DSNAME= & & dsname(PRIME). If you want to retrieve the temporary data set on the same job, you code only the data set's temporary name; you do not include the term PRIME, INDEX, or OVFLOW.

Associated Data Sets (3540 Diskette)

Associated data sets are data sets on 3540 diskette volumes that are separate from the job stream data set and are to be spooled as SYSIN data sets. Associated SYSIN data sets are identified by specifying a data set identifier (on the DD DSID parameter) and, optionally, a volume identifier on the DD * or DD DATA statements in the job stream.

To have associated data sets merged into the job stream, the job stream containing the diskette associated data set requests must be processed by the diskette reader program; it cannot be read by either JES2 or JES3.

Data sets are created on 3540 diskette volumes only by using SYSOUT. The SYSOUT DD statement must contain the DSID parameter and a sysout class (or classes) designed by the installation to be used by data sets on a 3540 diskette. The diskette writer must be started to the sysout class to transfer the data sets to diskettes.

For more information on the 3540 diskette, refer to **OS/VS2 IBM 3540 Programmer's Reference.**

Copying the Data Set Name from an Earlier DD Statement

The name of a data set that is used several times in a job, whether specified in the DSNAME parameter or assigned by the system, can be copied after its first use in the job. This allows you to easily change data sets from job to job and eliminates your having to assign names to temporary data sets. To copy a data set name, refer to an earlier DD statement that identifies the data set. Do not copy data set names of subsystem data sets created by a DD* or a DD DATA statement. When the earlier DD statement is contained in an earlier job step, you code DSNAME=*.stepname.ddname; when the earlier DD statement is contained in the same job step, you code DSNAME=*.ddname; when the earlier DD statement is contained in a cataloged procedure step called by an earlier job step, you code DSNAME=*.stepname.procstepname.ddname.

Specifying the DSNAME Parameter in Apostrophes

Sometimes, it may be necessary or desirable to specify a data set name that contains special characters. If the name contains special characters, you must enclose the name in apostrophes (5-8 punch), for example, DSNAME='DAT+5'. If one of the special characters is an apostrophe, you must identify it by coding two consecutive apostrophes (two 5-8 punches) in its place, for example, DSNAME='DAY"SEND'. A data set name enclosed in apostrophes can consist of 1 through 44 characters.

There are cases when the data set name must contain required special characters, which tell the system something about the data set (for example, & & in DSNAME= & & name are required special characters that tell the system that this is a temporary data set). In these cases, the data set name must not be enclosed in apostrophes because the system will not recognize the required special characters as having any special significance. The following data set names contain special characters that tell the system something about the data set and, therefore, cannot be enclosed in apostrophes:

- DSNAME=name(member name)
- DSNAME=name(area name)
- DSNAME=name(generation number)
- DSNAME= & & name
- DSNAME=*.stepname.ddname
- Part of, or the entire data set name, that is to be symbolically substituted

Keep the following rules in mind:

- If the data set name ends with a blank character, the blank is ignored.

- If the only special character is a period used to create a qualified data set name, a hyphen, or plus zero (0-12 punch), you need not enclose the data set name in apostrophes.

Specifying the LABEL Parameter

Labels are used by the operating system to identify volumes and the data sets they contain, and to store data set attributes. Data sets residing on magnetic tape volumes usually have data set labels. If data set labels are present, they precede each data set on the volume. Data sets residing on direct access volumes always have data set labels. These data set labels are contained in the volume table of contents of the direct access volume.

A data set label may be a standard or nonstandard label. Standard labels can be processed by the system; nonstandard labels must be processed by nonstandard label processing routines, which the installation includes in the system. Data sets on direct access volumes must have standard labels. Data sets on tape volumes usually have standard labels, but can have nonstandard labels or no labels.

The LABEL parameter must be coded if:

- You are processing a tape data set that is not the first data set on the reel; in this case, indicate the data set sequence number.
- The data set labels are not IBM standard labels; you must indicate the label type.
- You want to specify what type of labels a data set is to have when it is written on a scratch volume; indicate the label type.
- The data set is to be password protected; specify PASSWORD when creating the data set.
- The data set is to be processed only for input or output and this conflicts with the processing method indicated in the OPEN macro instruction; specify IN, for input, or OUT, for output.
- The data set is to be kept for a specific period of time; indicate a retention period (RETPD) or expiration data (EXPDT).

The Data Set Sequence Number Subparameter

When placing a data set on a tape volume that already contains one or more data sets, specify where the data set is to be placed, that is, whether the data set is to be the second, third, fourth, etc., data set on the volume. The data set sequence number causes the tape to be positioned properly so that the data set can be written on the tape or retrieved.

The data set sequence number subparameter is a positional subparameter and is the first subparameter that can be coded. The data set sequence number is a 1- to 4-digit number. The data set sequence number is ignored for the following types of data sets:

- For data sets passed from a previous step, the system obtains the data set sequence number from the passing step.
- For GDG ALL requests, the system always retrieves the data set sequence number from the catalog.

If you omit the data set sequence number subparameter or code 0, the system assumes 1 (this is the first data set on the tape) unless the data set is cataloged. If the data set is cataloged, the system obtains the data set sequence number from the catalog.

The Label Type Subparameter

The label type subparameter tells the system the type of labels associated with the data set. The label type subparameter is a positional subparameter and must be coded second, after the data set sequence number subparameter. You can omit this subparameter if the data set has IBM standard labels.

The label type subparameter is specified as:

- SL — if the data set has IBM standard labels.
- SUL — if the data set has both IBM standard and user labels.
- AL — if the data set has American National Standard labels.
- AUL — if the data set has American National Standard labels and American National Standard user labels.
- NSL — if the data set has nonstandard labels.
- NL — if the data set has no labels.
- BLP — if you want label processing bypassed.
- LTM — bypass leading tape mark, if encountered, on unlabeled tape. (OS/DOS interchange)

SL or SUL is the only label type that can be specified for data sets that reside on direct access volumes. SL, SUL, AL, AUL, NSL, and NL are the only label types that can be specified for data sets that reside on tape volumes. BLP and LTM are label type subparameters that can also be coded for tape.

When SL or SUL is specified, or the label type subparameter is omitted and the data set has IBM standard labels, the system can ensure that the correct tape or direct access volume is mounted. When specifying NSL, installation-provided nonstandard label processing routines must ensure that the correct tape volume is mounted. When specifying NL or BLP, the operator must ensure that the correct tape volume is mounted. If you specify NL, the data set must have no standard labels. When specifying AL or AUL, the system ensures that the correct American National Standard labeled tape is mounted.

For cataloged and passed data sets, label type information is not kept. Therefore, referring to a cataloged or passed data set that has other than standard labels, code the LABEL parameter and specify the label type.

BLP is not a label type, but a request that the system bypass label processing. This specification allows you to use a blank tape or overwrite a seven-track tape that differs from the current parity or density specifications. If the bypass label processing option is not selected by the installation and you have coded BLP, the system assumes NL.

Note for BLP: When requesting the system to bypass label processing and the tape volume has labels, the system treats anything between tapemarks as a data set. Therefore, in order for a tape with labels to be positioned properly, the data set sequence number subparameter of the LABEL parameter must be coded and the subparameter must reflect all labels and data sets that precede the desired data set. The **OS/VS Tape Labels** publication illustrates where tapemarks appear.

Nonspecific volume request: The label type subparameter can also be specified when making a nonspecific volume request for a tape volume (that is, no volume serial numbers are specified on the DD statement) and when having a certain type of labels. If the volume that is mounted does not have the corresponding label type desired, you may be able to change the label type.

When specifying NL or NSL and the operator mounts a tape volume that contains standard labels, you can use the volume provided: (1) the expiration data of the existing data set on the volume has passed; (2) the existing data set on the volume is not password protected; and (3) you make a nonspecific volume request. All of these conditions must be met. If they are not, the system requests the operator to mount another tape volume.

If you specify SL and make a nonspecific volume request, but the operator mounts a tape volume that contains other than IBM standard labels, the system asks the operator to identify the volume serial number and the volume's new owner before the IBM standard labels are written. If the tape volume has American National Standard labels, the system asks the operator for permission to destroy the labels. If you specify SL and make a specific volume

request, but the volume that is mounted does not contain IBM standard labels, the system rejects the tape and requests the operator to mount the tape volume specified.

The PASSWORD and NOPWREAD Subparameters

The PASSWORD and NOPWREAD subparameters tells the system that you want the data set to be password protected. If you specify PASSWORD, the data set cannot be read from, written into, or deleted by another job step or job unless the operator can supply the system with the correct password. If you specify NOPWREAD (no password read), the data set can be read without the operator supplying the password, but the password is still required for writing or deleting data sets.

The PASSWORD and NOPWREAD subparameters are positional and must be coded third, after the data set sequence number subparameter and the label type subparameter or the commas that indicate their absence. If you want the data set password protected, specify PASSWORD when the data set is created. Password protected data sets must have standard labels, either IBM standard or American National Standard labels.

The IN and OUT Subparameters

The basic sequential access method (BSAM) permits a specification of INOUT or OUTIN in the OPEN macro instruction as the processing method. If you have specified either of these processing methods in the OPEN macro instruction and want to override it, you may be able to do so by coding either the IN or OUT subparameter. For FORTRAN users, the IN and OUT subparameters specify whether the data set is for input or output.

When INOUT is specified in the OPEN macro instruction and you want the data set processed for input only, you can specify the IN subparameter. When the IN subparameter is coded, any attempt by the processing program to process the data set for output is treated as an error.

When OUTIN is specified in the OPEN macro instruction and you want the data set processed for output only, you can specify the OUT subparameter. When the OUT subparameter is coded, any attempt by the processing program to process the data set for input is treated as an error.

The IN and OUT subparameters are positional subparameters. If either is coded, it must appear as the fourth subparameter, after the data set sequence number subparameter, the label type subparameter, and the PASSWORD subparameter, or the commas that indicate their absence.

The RETPD and EXPDT Subparameters

When it is necessary that a data set be kept for some period of time you can tell the system how long it is to be kept when you create the data set. As long as the time period has not expired, a data set that resides on a direct access volume cannot be deleted by or overwritten by another job step or job, that has specified a disposition of DELETE for the data set. If it is necessary to delete a data set before the expiration date or retention period has passed use one of the following:

- The DELETE command as described in **OS/VS Access Method Services** for data sets cataloged in a VSAM catalog, which makes the space occupied by the data set available for reallocation.
- The IEHPROGM utility as described in **OS/VS2 MVS Utilities** to delete the catalog entry for data sets that are not cataloged in a VSAM catalog.
- The SCRATCH macro with the 'OVRD' parameter as described in **OS/VS2 System Programming Library: Data Management** to delete the data set control block; this makes the space occupied by that data set available for reallocation.

When the expiration date of a data set is the current date, the data set is considered expired and can be deleted or written over by another data set.

There are two different ways to specify a time period: (1) tell the system how many days you want the data set kept, the RETPD subparameter, or (2) tell the system the exact date after which the data set need not be kept, the EXPDT subparameter.

If you code the RETPD subparameter, you specify a 1- to 4-digit number, which represents the number of days the data set is to be kept. If you code the EXPDT subparameter, you specify a 2-digit year number and a 3-digit day number (for example, January 1 would be 001, July 1 would be 182), which represents the date after which the data set need not be kept. When neither the RETPD or EXPDT subparameter is specified for a new data set, the system assumes a retention period of zero days.

The RETPD or EXPDT subparameter must follow all other subparameters of the LABEL parameter. If no other subparameters are coded, you can code LABEL=RETPD=nnnn or LABEL=EXPDT=yyddd.

Example of Identifying Data Sets to the System

This job shows how to use the DSNAME parameter.

```
/*PRIORITY        8
//DATASETS      JOB     FREEMAN,MSGLEVEL=1
//STEP1         EXEC    PGM=IEFBR14
//D1            DD      DSN=ABC,DISP=(NEW,CATLG),UNIT=2314,
//                      VOL=SER=333001,SPACE=(CYL,(12,1,1),CONTIG)
//D2            DD      DSN=&&NAME,UNIT=3330,SPACE=(TRK,(10,1))
//D3            DD      DSN=SYSLIB,DISP=(OLD,KEEP)
//D4            DD      *
                .
                .
                .
                data
                .
                .
                .
/*
```

1. This job runs in priority 8, the meaning of which is defined by the installation.
2. The job statement specifies that system messages and JCL statements are to be printed.
3. D1 catalogs a newly created data set. The space request is 12 primary cylinders, 1 secondary, 1 directory, and the space is to be contiguous.
4. D2 creates a temporary data set on a 3330. The space request is for 10 primary tracks and 1 secondary.
5. D3 defines an old cataloged data set.
6. D4 defines a SYSIN data set. This will be followed by data in the input stream.

Disposition Processing of Non-VSAM Data Sets

Disposing of data sets at the end of a job step is known as disposition processing. You request disposition processing for non-VSAM data sets by coding the DISP parameter on the DD statement defining the data set. (VSAM data sets are handled differently. For information on VSAM, refer to **OS/VS2 Access Method Services.**) In the DISP parameter, you can code:

- Data set status as the first subparameter, indicating if the data set is new, is old, can be shared with other jobs, or can be lengthened.

- Normal disposition as the second subparameter, indicating how the data set should be handled if the job step terminates normally.
- Conditional disposition as the third subparameter, indicating how the data set should be handled if the job step terminates abnormally.

If you do not code one of the subparameters, or omit the DISP parameter entirely, the system supplies default values, as described under "Default Disposition Processing". Refer to Figure 25 for further information on disposition processing.

Specifying Data Set Status

Indicate a data set's status by coding one of the following:

- NEW — the data set is being created in this job step.
- OLD — the data set existed before this job step.
- SHR — the data set existed before this job step and can be read simultaneously by other jobs.
- MOD — the system first assumes that the data set exists. However, if the system cannot find volume information for the data set on the DD statement, in the catalog, or passed with the data set from a previous step, the system then assumes that the data set does not exist and the data set is created for the job step. Specifying MOD for a new sequential data set causes the read/write mechanism to be positioned after the last record in the data set each time it is opened for output when it is being created.

When coding SHR, you are requesting shared control of the data set and the job should be reading the data set only. When coding NEW, OLD, or MOD, you are requesting exclusive control of the data set. Shared and exclusive control are described in this chapter under "Insuring Data Set Integrity".

Specifying a Disposition for the Data Set

You can specify one disposition, called a normal disposition, to be used when the job step terminates normally (successfully) and another disposition, called the conditional disposition, to be used when the job step terminates abnormally.

For normal disposition, you can request as the second subparameter that the data set be:

- Deleted by coding DELETE.
- Kept by coding KEEP.
- Cataloged by coding CATLG.
- Uncataloged by coding UNCATLG.
- Passed by coding PASS.

Note: The disposition of a data set is solely a function of the DISP parameter; however, the disposition of the volumes on which the data set resides is a function of the volume status when the volume is demounted.

For conditional disposition (the third subparameter of the DISP parameter), you may code all of the above with the exception of PASS. You should consider using conditional disposition every time you create or use a data set. Conditional disposition can be used to keep data sets after a program failure, when they might be needed to determine the cause of the failure. Conditional disposition can also be used to delete data sets in case of program failure, thereby restoring the system environment to what it was before the error. This allows the failing job to be rerun without an intervening clean-up job.

Data sets allocated to steps that have abnormally terminated and that do not have automatic restart are disposed of as specified by the conditional disposition. If a job step abnormally

terminates during execution and a conditional disposition is not specified, the normal disposition is processed. If a job step fails during step allocation:

- A data set created in that job step is deleted.
- A data set that existed before that job step is kept.

Disposition processing differs for data sets on direct access volumes and data sets on magnetic tape volumes. A direct access volume contains a volume table of contents (VTOC) which consists of control blocks describing the non-VSAM data sets and available space on the volume. The handling of tape and direct access volumes when specifying a particular disposition is described below.

Deleting a Data Set

Specifying DELETE requests that the data set's space on the volume be released at the end of the job step (when coded as the normal disposition) or if the step abnormally terminates (when coded as the conditional disposition). If the data set resides on a public tape volume, the tape is rewound and the volume is available for use by other job steps. If the data set resides on a private volume, the tape is rewound and unloaded. In this case, it is rewound and unloaded and a KEEP message is issued. If the data set exists on a direct access volume, the control block describing the data set is removed from the VTOC and the space on the volume is then available to other data sets.

In one case, however, a data set on a direct access volume will not be deleted, even though you specify DELETE: when the expiration date or retention period has not expired. Specify a length of time that a data set must be kept by assigning a retention period or expiration date in the LABEL parameter on the DD statement. Specifying a retention period or expiration date is described under "Specifying the LABEL Parameter".

If you are deleting a cataloged non-VSAM data set, the entry for the data set in the system catalog is also removed, provided the system obtained volume information for the data set from the catalog (that is, the volume's serial number was not coded on the DD statement). If the system did not obtain volume information from the catalog, the data set is still deleted but its entry in the catalog remains. If an error is encountered while attempting to delete a data set, its entry in the catalog will not be removed. (The data set will or will not be deleted, depending on where the error occurs). To delete an entry from a VSAM catalog, use the DELETE command as described in **OS/VS2 Access Method Services**, which makes the space occupied by the data set available for reallocation. To delete the catalog entry for data sets that are not cataloged in a VSAM catalog, use the UNCATLG statement of IEHPROGM as described in **OS/VS2 MVS Utilities**.

DELETE is the only valid conditional disposition for a data set with no name or a temporary name. If a disposition other than DELETE is specified, the system assumes DELETE.

Keeping a Data Set

Specifying KEEP instructs the system to keep a data set intact until a subsequent job step or job requests that the data set be deleted or at least until the expiration date or retention period is passed. You can specify an expiration date or retention period, indicating the length of time a data set must be kept, in the LABEL parameter on the DD statement. If you do not specify a time period, the system assumes a retention period of zero days. Coding an expiration date or retention period is described under "Specifying the LABEL Parameter" in this publication.

If you are assigning a final disposition of KEEP to a passed data set, make certain that the rules for receiving a passed data set are followed. See the discussion under "Passing a Data Set" in this chapter.

For data sets on direct access devices, the entry describing the data set in the VTOC and the data set itself is kept intact. For data sets on tape, the volume is rewound and unloaded and a KEEP message is issued to the operator.

Cataloging a Data Set

Cataloging allows you to keep track of and retrieve data sets. Data sets can be cataloged in the system master catalog or in user (private) catalogs. When retrieving a cataloged data set, you do not have to specify volume information, you need only code the DSNAME parameter and a status in the DISP parameter other than NEW.

To catalog a non-VSAM data set, code CATLG as the disposition; the system creates an entry in the catalog that points to the data set. The disposition CATLG implies KEEP.

You can specify a disposition of CATLG for an already cataloged data set. This should be done when lengthening the data set with additional output (a status of MOD is coded) and the data set can exceed one volume. If the system obtained volume information for the data set from the catalog (that is, the volume's serial number was not coded on the DD statement) and you code DISP=(MOD,CATLG), the system updates the entry to include the volume serial numbers of any additional volumes.

A collection of cataloged data sets that are kept in chronological order can be defined as a generation data group (GDG). The entire GDG is stored under a single data set name; each data set within the group, called a generation data set, is associated with a generation number that indicates how far removed the data set is from the original generation. For more information on defining and creating generation data groups, see "Generation Data Groups" in this publication, and **OS/VS2 Access Method Services.**

Note: There are instances when a data set will not be cataloged. A data set will not be cataloged if the DD statement describing the data set is not opened by the problem program and

- You request a non-specific tape volume (scratch volume is assumed) or,
- You request a tape volume for a tape unit with dual density options and you did not specify the density (DEN subparameter of the DCB parameter) on the DD statement.

Uncataloging a Data Set

To remove the entry describing a non-VSAM data set from the catalog, code UNCATLG as the disposition. Specifying UNCATLG does not request the initiator to delete the data set — just the reference in the catalog is removed. When you request use of the data set in a subsequent job or job step, you must include volume information on the DD statement.

Passing a Data Set

If more than one step in a job requests the same data set, each step using the data set can pass the data set for use by a subsequent step. A data set can only be passed within a job.

To pass a data set, you code PASS as the normal disposition; PASS cannot be specified as the conditional disposition. You continue to code PASS each time the data set is referred to until the last time it is used in the job. At this time, you assign it a final disposition.

Specifying the data set name of a passed data set without specifying volume serial number or a volume reference is called "receiving" the data set. Identical data set names (whether or not the same data set is referred to) can be passed at the same time. Such identical data set names are received in the same order in which they are passed. A data set name that has been passed n times can be received no more than n times. A data set cannot be passed and received within the same step.

For a description on how the PASS subparameter affects volume disposition, see **OS/VS2 System Programming Library: Job Management.**

Considerations for Passed Data Sets - Consider the following when you pass data sets:

- A data set may be passed more times than it is received. However, a problem can occur when the same data set is passed more times than it is received in a procedure that is called multiple times in a job.

For example, the following procedure is called in a job step:

```
//STEP1     EXEC    PGM=IEFBR14
//DD1       DD      DSNAME=&A,DISP=(NEW,PASS),
//                  SPACE=(TRK,(1,1)),UNIT=SYSDA
//DD2       DD      DSNAME=*.DD1,DISP=(OLD,PASS),
//                  VOL=REF=*.DD1
//STEP2     EXEC    PGM=IEFBR14
//DD3       DD      DSNAME=&A,DISP=(OLD,DELETE)
```

In this example:

- DD1 and DD2 pass data set &A.
- DD3 receives data set &A.
- When the procedure is called the first time, one entry for data set &A remains unreceived.
- If the procedure is called a second time, DD3 receives data set &A from the first execution of the procedure and this can result in incorrect data or an abend.
- If data set &A is not received twice in the job, data set &A is processed as an unreceived passed data set at the end of the job.

- If a job step containing a passed data set abends during execution, the passed data set is passed at the end of the job step. This allows you to receive and process the passed data set on a following job step (for example, when COND=EVEN or ONLY is coded). If the passed data set remains unreceived at the end of the job, then the conditional disposition (if specified) for the passed data set occurs.

Disposition Processing of Passed Unreceived Data Sets

A data set can be passed by a job step and not subsequently received by another job step. If a job step abnormally terminates, unreceived data sets that specified a conditional disposition when passed are processed as specified in their conditional disposition, with four exceptions. If the conditional disposition requires an update to a user catalog:

- and CATLG is specified for a data set with a first-level qualifier of a catalog name or alias, the data set will not be cataloged.
- and UNCATLG or DELETE (of a cataloged data set) is specified for a data set with a first-level qualifier of a catalog name or alias, the data set will not be uncataloged.
- and CATLG is specified for a data set with no qualifier or with a qualifier that is not a catalog name, the data set will be cataloged in the master catalog.
- and UNCATLG or DELETE (of a cataloged data set) is specified for a data set with no qualifier or with a qualifier that is not a catalog name, an attempt will be made to uncatalog the data set from the master catalog.

Unreceived passed data sets that do not specify a conditional disposition, that is, those that were specified as (NEW,PASS) in this job, are deleted; all others are kept.

If unreceived passed data sets are deleted at the end of a job, dynamic allocation is performed to allocate the unit and volume for deletion.

If no job step abnormally terminates before it begins execution, unreceived passed data sets that were specified as (NEW,PASS) are deleted; other data sets are kept.

If a step abnormally terminates before it actually begins execution (for example, during allocation of units and volumes or direct access space), the system ignores the disposition you code and again automatically keeps existing data sets and deletes new data sets.

For example, if you code:

```
DISP=( ,PASS,CATLG )
```

the system assumes the data set is new. If any job step abnormally terminates during its execution, the system will catalog the data set, as instructed by the conditional disposition of CATLG. If, however, the step abnormally terminates before it actually begins execution, the system will delete the data set, since it is a new data set.

Default Disposition Processing

If you do not code the DISP parameter, or omit one of the subparameters, the system supplies default values.

If you do not specify a data set status, the system assumes NEW. If you do not code the second and third subparameters, the system determines how a data set should be handled according to the status of the data set: data sets that existed before the job are automatically kept (data sets for which OLD, SHR, or MOD is coded when volume information is available): data sets created in the job are automatically deleted (data sets for which you coded NEW or MOD when volume information is not available, or for which you did not code a status).

Bypassing Disposition Processing

If you define a data set as a dummy data set, the DISP parameter, if coded, is ignored and disposition processing is not performed. For details, see "Defining a Dummy Data Set."

Insuring Data Set Integrity

When a job must receive control of the data sets it requests, you can request either exclusive control, allowing no other job to use the data set, or shared control, allowing the data set to be used by other jobs that also request shared control. The process of securing control of data sets for use by a job is called data set integrity processing.

Data set integrity processing avoids conflict between two or more jobs that request use of the same data set. For example, two jobs, one named READ and another named MODIFY, both request the data set FILE. READ wants only to read and copy certain records, MODIFY deletes some records and changes other records in the data set FILE. If both jobs have control of FILE concurrently, READ cannot be certain of the records contained in FILE — cannot be sure of the integrity of the data set. MODIFY should have exclusive control of the data set; READ can share control of FILE with other jobs that also want only to read the data set. Indicate the type of control a data set requires in the DISP parameter on the DD statement defining the data set.

Exclusive Control of a Data Set

When a job has exclusive control of a data set, no other job can use that data set until termination of the last step in the job that refers to the data set. A job should have exclusive control of a data set in order to modify, add, or delete records.

In some cases, you may not need exclusive control of the entire data set. You can request exclusive control of a block of records by coding the DCB, READ, WRITE, and RELEX macro instructions. (These instructions are described in **OS/VS2 MVS Data Management Macro Instructions.**)

To request exclusive control of a data set, you code NEW, OLD, or MOD as the first sub parameter of the DISP parameter.

Shared Control of a Data Set

A data set on a direct access storage device can be used concurrently by several jobs, if these jobs request shared control of the data set; however, none of the jobs should change the data set in any way.

To request shared control, you code SHR as the first subparameter in the DISP parameter. If more than one step of your job requests a data set, you must code SHR every time you define the data set if it is to be used by concurrently executing jobs. Data set integrity processing is performed once for a job; a data set has either shared or exclusive control. If you code NEW, OLD, or MOD on any reference to a data set, the system assigns exclusive control to the data set for the entire job; a reference requesting exclusive control will override any number of references requesting shared control.

How the System Performs Data Set Integrity Processing

Data set integrity processing is performed for:

- Nontemporary data sets.
- Non-VIO temporary data sets (see "Using Virtual Input/Output (VIO) for Temporary Data Sets").
- Data sets with alias names (created with the Access Methods Services DEFINE command; see **OS/VS2 Access Methods Services.**)
- Members of generation data groups.

To secure control of a data set for a job, the system enqueues on the data set, marking the data set as requested by that job and noting what kind of control was requested. The job will receive control of the data set if:

- The data set is not being used by another job, or
- The data set is being used by another job but both the job requesting the data set and the job using the data set request shared control.

For example, a job named READ requests shared control of a data set named FILE; if FILE is being used by a job named LOOKAT and LOOKAT also requests shared control, both READ and LOOKAT can use the data set at the same time.

A job will not receive control of a data set if:

- The data set is being used by another job and that job has exclusive control, or
- The data set is being used by another job (with either exclusive or shared control), but the job requesting use of the data set requests exclusive control.

For example, the job named MODIFY requests exclusive control of the data set FILE; FILE is already being used by the job LOOKAT. MODIFY cannot receive control of the data set until LOOKAT has terminated.

If a job requests data sets that are not available, the system issues the message "JOB jjj WAITING FOR DATA SETS". The initiator that started the job will automatically wait until the required data sets become available, unless the operator cancels the job. However, a job will fail if it requests a data set with an alias name, or a member of a generation data group, and the data set is not immediately available.

Examples of Disposition Processing of Non-VSAM Data Sets

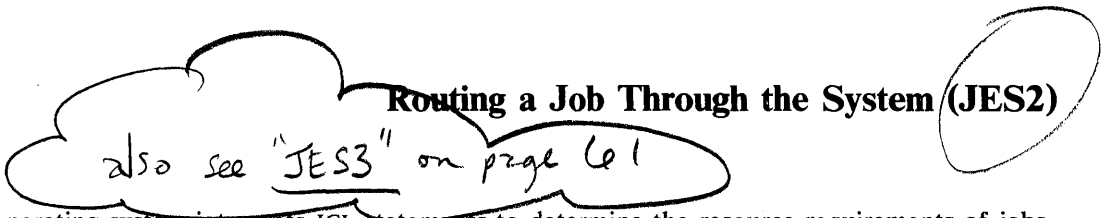
```
//DISP      JOB     MSGLEVEL=1
//S1        EXEC    PGM=IEFBR14
//D1        DD      DSN=ABC,DISP=(SHR,KEEP)
//D2        DD      DSN=SYSA,DISP=(OLD,DELETE,UNCATLG)
//D3        DD      DSN=SYSB,UNIT=2314,VOL=SER=231401,
//                  SPACE=(CYL,(4,2,1)),DISP=(NEW,KEEP,CATLG)
//D4        DD      DSN=&&SYS1,DISP=(MOD,PASS),UNIT=2314,
//                  VOL=SER=231404,SPACE=(TRK,(15,5,1))
//S2        EXEC    PGM=IEFBR14
//D1        DD      DSN=&&SYS1,DISP=(MOD,DELETE),UNIT=2314,
//                  VOL=SER=231404,SPACE=(TRK,(15,5,1))
```

1. The JOB statement requests that all JCL statements and system messages be printed.
2. D1 in S1 defines a data set that already exists and can be shared with other data sets. It is to be kept on the volume after this job step.
3. D2 in S1 defines a data set that already exists, cannot be shared with other data sets, is to be deleted at the end of the job step, and is to be uncataloged if the program abnormally terminates.
4. D3 in S1 defines a new data set that is to be assigned a specific volume (231401) on a 2314 device. The data set is to be kept on the volume at the end of this job step for normal processing but is to be cataloged if the program abnormally terminates.
5. D4 in S1 defines a temporary data set that is to be created in this job step. It is to be assigned to volume 231404 on a 2314 device with the space request of 15 primary tracks, five secondary, and a directory. This data set is to be passed for subsequent use by a job step in this job.
6. D1 in S2 defines the same temporary data set that was defined in D4 of S1. When this step is completed, the data set is to be deleted.

```
//PASS    JOB     MSGLEVEL=1
//S1      EXEC    PGM=IEFBR14
//DD1     DD      DSN=A,DISP=(NEW,PASS),VOL=SER=231400,
//                UNIT=2314,SPACE=(TRK,1)
//DD2     DD      DSN=A,DISP=(OLD,PASS),VOL=REF=*.DD1
//DD3     DD      DSN=B,DISP=(OLD,PASS),VOL=SER=231400,UNIT=2314
//DD4     DD      DSN=B,DISP=(OLD,PASS),VOL=SER=231401,UNIT=2314
//S2      EXEC    PGM=IEFBR14
//DD5     DD      DSN=A,DISP=OLD
//DD6     DD      DSN=A,DISP=OLD
//DD7     DD      DSN=B,DISP=OLD
//DD8     DD      DSN=B,DISP=(OLD,PASS)
//S3      EXEC    PGM=IEFBR14
//DD9     DD      DSN=B,DISP=OLD
```

1. DD1 and DD2 pass the same data set. DD5 and DD6 receive that same data set.
2. DD3 and DD4 pass different data sets of the same name, DD7 receives the data set passed by DD3 and DD8 receives the one passed by DD4. DD8 also continues to pass the data set originally passed by DD4.
3. DD9 receives the data set passed by DD4 and DD8.

Routing a Job Through the System (JES2)



The operating system interprets JCL statements to determine the resource requirements of jobs and job steps. The job entry subsystem 2 (JES2) reads a job into the system, satisfies the requirements requested on JCL and JES2 statements, schedules the job, and selects it for execution. JES2 automatically performs most of these services for you, but you can code JCL and JES2 parameters to influence how these services are performed. For example, JES2 schedules a job for execution, but you can influence when the job is selected by coding the JES2 PRIORITY control statement and the CLASS parameter of the JOB statement.

This section contains seven topics:

- Job Scheduling
- Passing Information to the Job in Execution
- Delaying Job Initiation
- Bypassing Job Initiation
- Conditional Execution of Job Steps
- Restarting a Job
- Example of Routing a Job Through the System (JES2)

Job Scheduling

JES2 controls the selection of jobs for processing. As JES2 reads a job into the system, JCL statements and any input stream data are placed on respective logical data sets. The JCL and JES2 statements are checked for syntax errors and appropriate error messages are issued. If the JCL statements are syntactically correct, the job is placed on an execution queue. The execution queue is divided into job class queues and, within each job class queue, jobs are placed according to their priority. A JES2 initiator is assigned job classes to process. It selects jobs from the first class assigned to it according to the priority of the jobs until no more jobs exist in that class, and then selects jobs from the next class assigned. The CLASS parameter on the JOB statement and the JES2 PRIORITY control statement can be used to influence how a job is placed on the execution queue. Caution should be taken in the shared spool environment as more than one JES2 system will be altering the queues. Due to conversion and CPU timings, jobs of the same class and priority may be queued for execution out of their reader sequence. Therefore, to insure that one job is selected before another or that the desired volumes are mounted before a job is executed, delay the job's selection by coding TYPRUN=HOLD on the JOB statement, by coding a job class that will force TYPRUN=HOLD, or by coding a SETUP control statement.

The job initiation stage can be entirely bypassed by coding TYPRUN=SCAN or TYPRUN=COPY on the JOB statement or by coding a job class that will force either of these options.

MVS includes support for controlling the processing rate of jobs and job steps. The installation defines a certain number of performance group definitions. Each of these defines a particular processing rate formula which is to be used for associated jobs or job steps. To associate a job or job step with performance group definitions, code the PERFORM parameter on either the JOB or EXEC statements.

Assigning a Job to a Job Class

Job classes are established by an installation to group jobs. By assigning jobs to job classes, the installation tries to avoid contention between jobs that require the same resources by

preventing them from running concurrently and tries to provide a better mix of jobs for more efficient system use. The installation determines which characteristics are most important in achieving a good balance of jobs in the computing system. Assign a job to a job class by coding the CLASS parameter on the JOB statement.

Assigning a Priority to a Job

Within a job class, jobs are selected for execution from the execution queue according to job priority. Jobs with the same class and priority are placed in the execution queue in a first in/first out order. In most cases, JES2 will calculate the job's priority. However, for certain jobs, you or the operator can be instructed to assign different priorities. Specify job priority by coding a JES2 PRIORITY statement or by coding the PRTY parameter on the JOB statement.

Priority is explicitly stated on a PRIORITY statement and used by JES2. The estimated number of cards, lines of output, and the time for job execution are used according to installation algorithms to calculate the priority, and are also used by JES2 to monitor job execution time and output. If these estimates are not stated, installation defaults are assumed. If any of these estimates are exceeded, the operator is notified. In some cases, the installation can specify that the job be canceled. For example, an installation might specify that the lower the estimated execution time and output, the higher the priority. This can enforce correct amounts to be specified or the job will be canceled. JES2 sets a job's selection priority to 1 following execution if the job's execution priority was 12 or less. If the execution priority was greater than 12, JES2 sets the selection priority to 15.

Assigning a Dispatching Priority to Job Steps

In most jobs, you will want the job's dispatching priority to default to an automatic priority group (APG) instead of assigning your own dispatching priority. The automatic priority group function is an algorithm that the system resources manager will use to attempt to increase system throughput by dynamically adjusting the dispatching priority of associated address spaces.

If you do assign a dispatching priority, code the DPRTY parameter on the EXEC statement. In the DPRTY parameter, you can code two values. The system substitutes these values in the following formula to form the dispatching priority:

(value1 x 16) + value2 = step's dispatching priority

If you omit the DPRTY parameter completely, the job step is assigned the APG priority. If value1 is omitted or it is equal to the APG value, the step is assigned the APG priority and any value you code for value2 is ignored. In this case, value2 is obtained from the Installation Performance Specification (IPS) using the performance group associated with the job step. (See **OS/VS2 System Programming Library: Initialization and Tuning Guide** for information on IPS.) If value2 is not specified in the IPS, a value of 6 is assigned to value2.

Performance of Jobs and Job Steps

You can associate a job or job step with any one of several performance group definitions. Performance group definitions which are supplied by the installation, describe the workload-dependent processing rate which should be afforded to an associated job or job step. Most performance group definitions prescribe good processing rates under light system workload conditions. However, when the system workload is moderate or heavy, some performance group definitions will specify significantly lower processing rates than for other performance groups. The installation defines the number and definition of performance groups needed to meet the response requirements of its various users and will probably publish this

information for your use. Make the performance group association with the job or job step by coding an appropriate performance group number on the PERFORM parameter of the JOB or EXEC statement.

For further information concerning performance, refer to **OS/VS2 System Programming Library: Initialization and Tuning Guide** and to **System Programming Library: JES2 Installation, Initialization and Tuning.**

Routing a Job in a Network (JES2)

JES2 enables users to enter a job on one system and have it executed on another system and to send output from one system to a remote device or an output device on another system. Jobs can be routed in two ways:

- An operator may issue a JES2 command through a console or input reader, (refer to **Operator's Library: JES2 Commands.**
- JES2 control statements can be specified within the JCL.
  Execution routing through JCL may be accomplished with either of two control statements:
- /*ROUTE: Includes an option, XEQ. Specifying the XEQ option with the name of a node sends the job to that node for execution.
- /*XEQ: Includes an operand for which the user need only specify the node name to send the job to that node for execution.

The job may be entered at an RJE terminal, a TSO terminal, an input reader, or an internal reader before being routed to some other node for execution. A started task or a TSO logon, however, cannot be executed at any node other than the node of entry.

Transmitting Data in a Network (JES2)

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

Passing Information to the Job in Execution

Some information required by a program can vary from application to application, such as module attributes and options required by the compiler, assembler, and linkage editor programs. In order to provide this information to the program at the time it is executed, code the PARM parameter on the EXEC statement. The program must include instructions that can retrieve this information. (The exact location and format of the information passed to a processing program are included in **OS/VS2 Supervisor Services and Macro Instructions.**)

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

Identifying the Program to be Executed

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

Temporary Library

If in a job you want to assemble, linkage edit, and then execute a program, make the output of the linkage editor a member of a partitioned data set. This is accomplished by creating a temporary library. A temporary library is a partitioned data set created in the job to store a program as a member of the data set until it is executed in a subsequent job step. When the

program is required, refer back to the DD statement that defines the temporary library and the member by code PGM=*.stepname.ddname or PGM=*.stepname.procstepname.ddname. You can also request use of a program that is a member of a temporary library by coding PGM=program name and including a DD statement named JOBLIB or STEPLIB that defines the temporary library. To keep this program available for use by other jobs, make the program a member of a private library.

Private Library

Request use of a program that is a member of a private library by coding PGM=program name and including a DD statement named JOBLIB or STEPLIB that defines the private library. The system automatically looks in the private library for a member with the corresponding name.

A program that resides in a private library can also be executed by coding PGM=*.stepname.ddname or PGM=*.stepname.procstepname.ddname. This can be done only when the named DD statement defines the program as a member of a private library.

The IEFBR14 Program

This program is used to check the syntax of the control statements, allocate space, or satisfy requests for disposition processing prior to execution of a job. To do this, substitute IEFBR14 for the program name on the EXEC statement.

If you created a data set when using this program, the data set's status will be old when you execute your program. For situations when a disposition of CATLG will not be satisfied see "Cataloging a Data Set" in this publication.

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

Selecting a Cataloged Procedure Library

You can choose which of the installation specified cataloged procedure libraries will be used for resolving catalog procedure references in the JCL by coding the PROCLIB parameter on the JOBPARM statement. If this parameter is omitted, a cataloged procedure library associated with your job's class will be used.

Delaying Job Initiation

Although you can influence a job's selection by assigning a job class and priority to the job, you cannot predict whether a job in one job class queue will be selected for execution before another job in a different job class queue. When jobs exist in the same job class queue, you cannot be certain that one job will complete execution before the other job is selected, even if you assign a higher priority to the first job. In some cases, when submitting two jobs, JOBA and JOBB, JOBA must complete execution before JOBB is initiated—JOBA might create records that JOBB will use. JOBB's initiation will have to be delayed until JOBA completes execution. It is also possible that resources a job requires will not be available—in this case, you will want to delay the job's initiation until required resources are available. Delay a job's initiation by coding TYPRUN=HOLD or TYPRUN=JCLHOLD on the JOB statement or by specifying a job class defined by the installation to force a TYPRUN=HOLD default. You can also delay a job's initiation and have specific volumes mounted before the job executes, by coding the SETUP control statement to notify the operator which volumes are required.

To delay a job's initiation, code SETUP, TYPRUN=HOLD, TYPRUN=JCLHOLD, or a held job class; the job remains on the execution queue or JCL conversion queue but cannot be selected for processing until the operator releases the job. When the operator releases the job, it is again eligible for selection.

If you code a SETUP control statement, you are able to notify the operator what volumes are to be retrieved from the library. The operator will mount the requested volumes and then should release the job which has been held on the execution queue or on the JCL Conversion Queue.

Bypassing Job Initiation

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

To scan the control statements for syntax errors without initiating the job, code TYPRUN=SCAN on the JOB statement or select a job class which the installation has defined to force the TYPRUN=SCAN default. With this option coded, the job is first scanned for control statement syntax errors and then passed directly to the output stage for processing.

To produce a copy of the input deck without initiating any steps, code TYPRUN=COPY on the JOB statement or select a job class which the installation has defined to force the TYPRUN=COPY default. With this option coded, the input deck (as submitted) is converted directly to a SYSOUT data set and scheduled for output processing. The class of the SYSOUT data set is the same as the message class of the job and can be controlled by the MSGCLASS parameter on the JOB statement. The SYSOUT data set generated can be processed by either the JES2 output processor or by the external writer, but is not available to the TSO OUTPUT command. JES2 control statements encountered in the input stream are interpreted before being added to the SYSOUT data set; job control language (JCL) statements are copied without any processing (that is, no JCL conversion).

Conditional Execution of Job Steps (JES2) also see p. 73

Depending on the results of one step of a job, you may not wish to execute subsequent steps—if a compilation fails, you would not want to waste computing time attempting subsequent link-editing or execution steps. Specify tests to determine whether to bypass or execute job steps, based on the results from previous steps by coding the COND parameter on either a JOB or EXEC statement.

The results of a job step can be reflected in a **return code**, a number from 0 to 4095. The COND parameter can be coded to test the return codes which are issued by the compiler, assembler, and linkage editor programs. Some return codes are standard for certain programs; for example, a return code of 8 issued by a compiler or linkage editor indicates that serious errors were found and execution is likely to fail. In problem programs, assign a number as the return code to signify a certain condition. For example, if STEP1 of a job reads accounts to be processed in subsequent job steps, you might set a return code of 10 if no delinquent accounts are found. Before STEP3 is executed to process delinquent accounts, test the return code from STEP1; if the return code from STEP1 is 10—there are no delinquent accounts—you can skip STEP3. Specify the test to check the return code from STEP1 by coding the COND parameter.

You can also instruct the system to execute a step **even** if a previous step has abnormally terminated or **only** if a previous step has abnormally terminated by coding EVEN or ONLY inprev the COND parameter on a EXEC statement. For example, STEP1 of a job updates records in a data set. If STEP1 abnormally terminates, you want to execute STEP2, which will print the data set. Specify that STEP2 should be executed **only** if STEP1 abnormally terminates by coding ONLY in the COND parameter on the EXEC statement for STEP2.

Specifying Return Code Tests

In the COND parameter, specify tests to determine if the system should bypass a job step. If the system determines that a comparison is true, the job step is skipped (if COND was coded on the EXEC statement) or all remaining job steps are skipped (if COND was coded on the JOB statement).

For example, if you code COND=((10,GT),(20,LT)) on the EXEC statement, you are asking, "Is 10 greater than the return code or is 20 less than the return code?" If either is true, the job step is skipped.

If the return code is 12, neither test is satisfied: no job step is skipped. All the tests you specify must be false if processing is to continue without skipping any job steps. If the return code is 25, the first test is still false, but the second test is satisfied: 20 is less than 25. The system will bypass one job step or all remaining job steps, depending on if the COND parameter was coded on the EXEC statement or on the JOB statement.

Restarting a Job

When a job step abnormally terminates, you may have to resubmit the job for execution; this means lost computer time and a delay in obtaining the desired results. The operating system provides restart facilities to reduce the effects of abnormal termination.

If you have installed MVS/System Product-JES2 Release 2 (5740-XYS) without installing Data Facility/Device Support Release 1 Enhancements (5740-AM7) do not use the checkpoint/restart facility.

If you have installed Data Facility/Device Support with MVS/System Product you can use the checkpoint/restart facility with certain restrictions.

For additional information concerning these restrictions see **OS/VS2 MVS Checkpoint/Restart** and **OS/VS2 MVS System Programming Library: Job Management.**

There are two types of restarts:

- **Step restart**, from the beginning of a job step.
- **Checkpoint restart**, from a checkpoint within a job step. (You establish checkpoints in a job step by coding the CHKPT macro instruction for each checkpoint. The CHKPT macro is described in **OS/VS2 MVS Data Management Macro Instructions.**) See also the DD CHKPT parameter. It specifies that checkpoints are to be taken at end of volume for the data set defined by the DD statement on which it is coded.

Whether using step restart or checkpoint restart, the restart facility can be automatic or deferred.

Automatic restart: To use automatic restart, code the RD (restart definition) parameter on the JOB or EXEC statement. If you use this facility, the presence of a job journal is required. (A **job journal** is established at JES2 initialization in order to hold restart information for each program in execution.) When a system failure occurs or a job step abnormally terminates and you have a job journal, the restart facility allows you to have automatic restart by coding RD=R on the JOB or EXEC statements. If you have taken checkpoints, automatically restart at the last checkpoint regardless of whether you have coded the RD parameter. When a job step abnormally terminates or a system failure occurs while the job is in execution and you do not have a job journal, these jobs are ineligible for automatic restart regardless of whether or not the RD parameter is coded.

Deferred restart: To use deferred restart, code the RESTART parameter on the JOB statement. This required parameter specifies a job step or a step of a cataloged procedure and can specify a checkpoint identifier if you are using deferred checkpoint restart. The effect of the parameter

is simply to restart the job at the beginning of the specified step or checkpoint. The SYSCHK DD statement is required when a job is being submitted for deferred checkpoint restart and must specify explicit UNIT and VOLUME information if the checkpoint data set is not cataloged.

Refer to **OS/VS2 MVS Checkpoint/Restart** for a complete description of planning for and using the checkpoint restart facility.

Example of Routing a Job Through the System

The purpose of this job is to execute five steps to perform an unspecified function. Not all of the steps will execute because conditions are placed on them.

```
/*PRIORITY    *
//ROUTE      JOB      (D58706),ROEGER,MSGLEVEL=(1,1),CLASS=E
//STEP1      EXEC     PGM=IEFBR14
//DD1        DD       SYSOUT=A
//STEP2      EXEC     PGM=IEFBR14,COND=EVEN
//DD2        DD       SYSOUT=A
//STEP3      EXEC     PGM=IEFBR14,COND=ONLY
//DD3        DD       SYSOUT=A
//STEP4      EXEC     PGM=ABEND806
//DD4        DD       SYSOUT=A
//STEP5      EXEC     PGM=IEFBR14,COND=ONLY
//DD5        DD       SYSOUT=A
```

1. This job will use the installation-defined priority default.
2. The JOB statement specifies that only JCL statements and messages are to be written, that the job is assigned to job class E.
3. All SYSOUT data sets will be processed by output class A.
4. STEP1 will execute normally.
5. STEP2 will execute normally.
6. STEP3 will not execute.
7. STEP4 will execute and will abnormally terminate.
8. STEP5 will execute because a preceding step did abnormally terminate.

By coding JCL statements, you can request output data sets, listings of JCL statements, system messages, and abnormal termination dumps. By coding JCL and JES2 statements, you can request special forms processing, routing of output to specific devices, and multiple original printing by data sets within a job. The JES2 statements have the same options as JCL with some additional features such as multiple destination, left and right indexing feature for the 3211 printer, and data set grouping.

This section includes five topics:

- Requesting Listings of JCL Statements and System Messages
- Requesting an Abnormal Termination Dump
- Writing Output Data Sets
- Controlling Output Destination
- Example of Obtaining Output JES2

Requesting Listings of JCL Statements and System Messages

MVS produces messages about a job concerning allocation of units and volumes, disposition of data sets, and termination of job steps and the job. You can request that these messages and/or the JCL statements from the job and from cataloged procedures called by the job be included on an output listing.

By coding the MSGLEVEL parameter on the JOB statement, you inform the system of what statements and messages you want included on the output listing. (The notation used on the output listing to identify cataloged and in-stream procedure statements is described in the chapter "Using Cataloged and In-stream Procedures.")

By coding the MSGCLASS parameter on the JOB statement, you assign messages and JCL statements to an output class. A default is assigned if MSGCLASS is not coded.

Requesting an Abnormal Termination Dump

To obtain a dump in the event of abnormal termination of a job step, code a DD statement defining a dump data set. The name of the DD statement must be either SYSABEND, SYSMDUMP, or SYSUDUMP. If more than one of these DD statements is present, the system uses the last one. If you override a DD statement in a cataloged procedure the overriding DD statement is used because it is the last one encountered.

SYSABEND, SYSMDUMP, and SYSUDUMP DD statements can provide a dump containing the processing program's virtual storage areas, the system nucleus, the entire system queue area, all local system queue areas, and any active link pack area (LPA) modules for the failing task. If either the Generalized Trace Facility (GTF) or the System Trace is active, the dump will contain their records. (Both cannot be active at the same time and GTF has priority.) In addition, if your installation permits dumping of the common storage area, a SYSMDUMP DD statement can provide a dump containing those parts of the CSA that the failing program is authorized to reference.

You can take subsequent SYSMDUMPs to the same data set if the data set name is SYS1.SYSMDPxx (where xx can be 00 - FF) and the data set disposition is SHR. Before attempting to take subsequent SYSUDUMPs see "Rules for Coding" under "The SYSABEND, SYSMDUMP and SYSUDUMP Facilities" in this publication.

If your program has issued an ABEND macro, or if you have written a recovery routine, you can determine what dump options you want, in addition to the installation defaults, and define them in a dump option list on the ABEND macro or on a SETRP macro issued by your recovery routine. How to do this is explained in **OS/VS2 Supervisor Services and Macro Instructions.**

Dumps with more data per page are available with the 3800 Printing Subsystem. By specifying CHARS=DUMP on the related DD statement, the dump is formatted in a 204-character line containing 64 bytes of storage. If FCB=STD3 is specified, the page is printed at 8 lines per inch. The dump program recognizes only STD3 for producing 8 lines per inch.

Descriptions of dumps and information on reading dumps are included in the **OS/VS2 System Programming Library: Debugging Handbook** and **OS/VS2 System Programming Library: Diagnostic Techniques.**

To have the dump which was produced for a SYSABEND or SYSUDUMP DD statement printed, either assign the dump to an output class using the SYSOUT parameter on the DD statement, or code the UNIT parameter and specify the printer on which you want the dump printed. To store the dump, define the data set as you would any other data set, coding the DSNAME, DISP, UNIT, and VOLUME parameters. If the data set will go to a direct access device, code the SPACE parameter.

The dump taken for a SYSMDUMP DD statement is machine-readable (unformatted) and must be stored on either a magnetic tape unit or a direct access device. If the job or step is running with nonpageable virtual storage (ADDRSPC=REAL on the JOB or EXEC statements), the SYSMDUMP output must be directed to a VIO data set.

To format and print a dump taken for a SYSMDUMP DD statement, use the AMDPRDMP service aid which is documented in **OS/VS2 System Programming Library: Service Aids,** or IPCS which is documented in **OS/VS2 MVS Interactive Problem Control System (IPCS) System Information.** Do not print the dump by sending it to SYSOUT=A because the output will be unformatted and difficult to read.

If you are using IPCS to format and print a dump taken for a SYSMDUMP DD statement, the data set disposition specified will produce the following results:

- DISP=MOD on a SYSMDUMP DD statement permits the collection of dump information pertaining to each of multiple events that occur during one job step.

Use of DISP=MOD produces a data set which can only be processed by IPCS (although IPCS's COPYDUMP subcommand can generate AMDPRDMP input from any dump collected in such a data set).

- DISP=NEW or DISP=OLD on a SYSMDUMP DD statement permits the collection of only the dump information pertaining to the last of multiple events which occur during one job step.

Use of either DISP=NEW or DISP=OLD produces a data set that can be processed by either IPCS or AMDPRDMP.

If a private data set is specified for SYSABEND or SYSUDUMP and more than one dump is possible, the data set should be specified with a disposition of MOD as it will be closed after each dump.

Writing Output Data Sets

There are two ways to write output data sets:

- Assign the data set to an output class.
- Specify the device on which the output should be written.

When you assign a data set to an output class, it is handled by JES2. The data set is first written to the JES2 spool device and then written to the final output device by either JES2 or an external writer. When you specify the device on the UNIT parameter, if the device is available, it is exclusively assigned to your job and under the control of your program.

Output data sets to be written to a 3540 diskette must be assigned to an output class that is processed by the diskette writer (an external writer) as described in **OS/VS2 IBM 3540 Programmer's Reference.**

Assigning Output Data Sets to Output Classes

Output classes include output with similar characteristics that are written to the same device. There are 36 possible output classes that can be coded on either the SYSOUT or MSGCLASS parameters. The letter and number names have no inherent meaning;—each installation defines its own output classes. For example, output class W might contain low priority output; class X might be reserved for high-volume output. If you want the output data set and messages from

the job to be printed on the same output listing, specify SYSOUT=* or the same output class in the SYSOUT parameter as specified for messages in the MSGCLASS parameter.

JES2 assigns output priorities based upon the amount of output for a specific type and class. Thus, a job with 100,000 lines of printer output for one FORM type and 5,000 lines of output for another FORM type will have different priorities assigned to its two different outputs. Total assigned priority is the sum of the priority assigned according to the output volume and the job selection priority. For additional information about the processing of output priorities by JES2, see **System Programming Library: JES2 Installation, Initialization, and Tuning.**

The installation can designate certain SYSOUT data sets as reserved. Reserved classes can cause the data sets to be held; that is, not sent to JES2 output processing. If the output class specified for the MSGCLASS parameter is not designated as a reserved class, it will not be held and none of the job's data sets assigned to reserved classes will be held. Data sets can be explicitly held by coding the HOLD=YES parameter or by coding TSO commands. (Refer to **OS/VS2 TSO Command Language Reference** for information on the TSO commands.) Jobs can be released from the hold state by the operator or by the time-sharing user with the TSO OUTPUT command. By using reserved classes, the controlling of the holding or not holding of all desired print data sets is done by means of the MSGCLASS parameter on the JOB statement.

Specifying the Device

To write an output data set without using the JES2 SYSOUT service, code the UNIT parameter on the DD statement defining the device on which the data set is written. The system will allocate the device exclusively to the job if the device is available: no other job can write output to that device until it is released. Jobs cannot share an output device as they can when output is assigned to output classes.

Data management routines write the output from the program to the device specified in the UNIT parameter. Specifying a particular output device in the UNIT parameter generally is not the most efficient method for obtaining system output.

Processing Output Classes

Using JES2 is an efficient way to write output. JES2 supports the use of local and remote printers and punches as devices on which output classes are written. An external writer supports tape and direct access devices and user-written writer routines.

Job related output is output that is neither held nor spun off nor processed by a user-written writer. (A spun off data set is made available for output processing before job termination.) Job related output will be retained until the end of the job and printed by JES2. If you release a held data set in time for it to be printed with its job-related data sets, JES2 will print them together if the following criteria are satisfied:

- The released data set has not been spun-off; spin-off data sets are always printed separately.
- Printing of the job-related data set has not begun; released data sets are printed independent of job related data sets.
- The job-related data set does not have mulitiple copies; released data sets will not be printed with job-related data sets for which there are multiple copies.
- The released data sets would have been part of the job-related data set had they not been held.

Dynamically deallocated SYSOUT data sets are spun off and are not considered part of the job related output.

Output will be printed on the same listing if such parameters as CLASS, FORMS, FCB, UCS, and DEST have similar characteristics for all data sets and a user-written writer is not specified.

The installation may choose to put all data sets that specify the same output class as the MSGCLASS parameter out on the same listing, even though FORMS, UCS, FCB, and DEST are different.

For an external writer, the operator will determine what data set will be selected. This can cause certain output to print out on the same listing even though all of the FORMS, DEST, UCS, and FCB parameters do not indicate the same characteristics.

Either an IBM-supplied external writer or a user-written writer can process the output. The external writer must be started by the operator to have the data written to an output device. If you want to know more about how to write an external writer routine, refer to **OS/VS2 System Programming Library: Job Management.**

Delaying the Writing of an Output Data Set

Data sets can be delayed from normal printing or delayed for inspection from a time sharing terminal prior to actually printing on that terminal by specifying reserved classes and by coding the HOLD parameter. For example, the installation can direct the delayed printing of a very large data set to prevent monopolizing an output device until smaller data sets are written. If a data set requires special forms that are not immediately available, it can be held until the operator supplies those forms. When HOLD=YES is specified on the DD statement, the data set is placed on a hold queue until the operator releases it. Notify the operator (using the NOTIFY parameter for TSO or the MESSAGE statement for JES2) when that data set is ready for processing because no message will be sent to the operator. The data set can be released by the operator or time-sharing user for printing.

Suppressing the Writing of an Output Data Set

Whether writing an output data set by coding the SYSOUT parameter or the UNIT parameter, you can suppress the writing of the data set by defining it as a dummy data set. This is useful when testing a program and you do not want data sets printed until you are sure they will contain meaningful output. Suppressing the writing of a data set saves processing time.

If you are routing an output data set by coding the SYSOUT parameter, code the DUMMY parameter to define the data set as a dummy data set. When DUMMY is coded, the SYSOUT parameter is ignored and the data set is not written.

You can also suppress the writing of an output data set by specifying a particular installation-defined class defined to delete the data set before it is printed. This technique is used by the installation to suppress the output of started tasks such as START and MOUNT commands.

If the device on which the data set will be written is specified in the UNIT parameter, you can assign the data set a dummy status by coding DUMMY or by assigning the data set name NULLFILE. All parameters other than DUMMY or DSNAME=NULLFILE and DCB are ignored; no units are assigned to the data set. When the program requests that the data set be written, the request is recognized but no data is transmitted. Requests to write a dummy data set are supported by the basic sequential access method (BSAM), virtual storage access method (VSAM), and queued sequential access method (QSAM). If any other access method is used, the job is terminated.

Limiting Output Records

The number of logical records in the output data set can be limited by specifying a maximum number of records through the use of the OUTLIM parameter. For example, a program is printing and goes into an endless loop. You can anticipate this problem and only have a maximum number of records printed before having the system discontinue the output processing.

Requesting Page Overflow Processing

JES2 will automatically limit the number of lines printed per page, thus preventing printing over the edge of the form, if requested either by the installation during JES2 generation or by the programmer coding LINECT or the JOBPARM statement. The installation specified number of

lines per page can be overridden by the JOBPARM LINECT parameter or line limiting can be turned off by coding LINECT=0. Set the line count sufficiently large to prevent unwanted page ejects for output from programs that provide page eject carriage control parameters.

Interpretation of Punched Output

Cards punched on a 3525 card punch from output spooled by JES2 will be interpreted if you code FUNC=I as a DCB subparameter on the SYSOUT card and if the spooled output is processed by a JES2 writer rather than the external writer. The FUNC=I subparameter will be ignored if the spooled output is processed by the JES2 writer onto a card punch other than the 3525. You could check with the installation to determine if a special output class has been set aside for 3525 output. Card interpretation by the external writer is an operator specified function. Output to be interpreted should be placed in a class designated by the installation as a punch with interpretation class.

JES2 Support of the 3211 Indexing Feature

You can specify that output that is printed by the JES2 writer onto a 3211 printer be indexed to the right or to the left by coding the INDEX or LINDEX parameters, respectively, on the OUTPUT statement. These parameters will be ignored if the output is processed by the external writer or is processed to a device other than a 3211. Determine whether an output class has been set aside to designate output to be processed by a JES2 writer onto a 3211 printer by asking the installation's system programming staff.

Requesting Multiple Copies of an Output Data Set

You can control the number of hard copies produced by the SYSOUT data sets. As many as 255 copies of an output data set are obtained by coding the COPIES parameter on the SYSOUT DD statement defining the data set or on the JES2 OUTPUT control statement. As many as 255 copies of the entire job related output are obtained by coding the COPIES parameter on the JES2 JOBPARM control statement.

If you request multiple copies of job related output by coding the OUTPUT or SYSOUT DD statements and the JOBPARM control statement, JES2 output processing gives the multiple of the requested amount for each SYSOUT data set. For example, if you request two copies of the entire job output (code COPIES=2 on the JOBPARM statement) and three copies of a certain output data set (code COPIES=3 on a SYSOUT DD statement or OUTPUT control statement), you will receive two copies of the entire job output but will receive a total of six copies of the SYSOUT data set. If the data set has been written directly to an output device, held, spun off, or processed by an external writer, however, it is no longer a job related data set and is not affected by the COPIES parameter on the JOBPARM statement. In this example, you would receive three copies of the requested output data set.

With the 3800 Printing Subsystem, you can also specify on the SYSOUT DD statement how the copies of the output data set are to be grouped. Each group value of the COPIES parameter specifies the number of copies of each individual page that is to be printed before copies of the next page are printed. The total number of copies printed equals the sum total of the group values. Group values can also be coded on the COPYG parameter of the JES2 OUTPUT statement.

Requesting Copy Modification

Selected copies of output can be modified when using the 3800 Printing Subsystem by specifying a copy modification module name on the MODIFY parameter of the SYSOUT or output DD statement or on the JES2 OUTPUT statement. This allows the printing of predefined data on all pages of a specific copy or copies of a data set. For example, you may want to vary column headings or explanatory remarks on different copies of the same printed page of data. Copies might also be personalized with the recipient's name, address, and other desired information. Blanks or printable characters such as asterisks, might also be used to suppress the printing of variable data on particular copies of a page. (This is a function done in other printers by using short or spot carbon in the forms set.)

The predefined data is created as a copy modification module and stored on SYS1.IMAGELIB using the IEBIMAGE utility program. For information on using IEBIMAGE, see the **IBM 3800 Printing Subsystem Programmer's Guide.**

Requesting Printer Form and Character Control

When requesting that an output data set be printed, you can give JES2 special instructions on how to print the data set. You can request:

- A special output form.
- A special character set or arrangement, when output is being printed by a 3211, 3203-5, or 1403 printer with the universal character set feature or by a 3800 Printing Subsystem.
- A specific FCB (forms control buffer) module, which controls how many lines per inch are printed and the length of the form, when the data set is written to a 3211 or 3203-5 printer or a 3800 Printing Subsystem.
- A specific carriage control tape, when the data set is written to a 1403 printer.

Note: JES2 treats the 3203-5 printer the same as a 3211 printer with the following exceptions:

- The UCSs for the 3203-5 are the same as for the 1403 printer.
- The 3203-5 printer does not support indexing; therefore JES2 indexing commands are ignored.
- You can not explicity identify the 3203-5 printer to JES2 via JES2 initialization parameters (MVS passes the 3203-5 identification to JES2 via the UCB).
  For further information on UCSs and UCBs, see **OS/VS2 System Programming Library: Data Management.**

Requesting a Special Output Form

Special forms are requested for output data set printing by including the form name in the SYSOUT parameter on the DD statement defining the data set or by coding the FORMS parameter on the OUTPUT control statement. For example, assign a data set to an output class to be routed to a printer and specify the data set be printed on a special form. (For example, code SYSOUT=(A,,FMS2).) JES2 and the external writer insure that the proper form is mounted.

The entire job can be printed on a special form by coding the FORMS parameter on the JOBPARM statement. If you code a forms name on either the SYSOUT or the OUTPUT statements, it overrides the forms name in the JOBPARM statement.

Requesting a Special Character Set Using the UCS Feature

The universal character set (UCS) feature is requested by coding the UCS parameter on a DD statement defining an output or SYSOUT data set or by coding UCS on the OUTPUT control statement for SYSOUT data sets. You can request the UCS feature for different sets of characters to be printed for various applications.

To request a special character set for a 3211, 3203-5, or 1403 printer, specify the code identifying the character set in the UCS parameter or the OUTPUT statement. The codes for the IBM standard special character sets are in Figure 10.

1403	3203-5	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 alphameric character set, arrangement A
PCHN	PCHN		Preferred alphameric character set, arrangement H
PN	PN	P11	PL/I alphameric character set
QN	QN		PL/I preferred alphameric character set for scientific applications
QNC	QNC		PL/1 preferred alphameric characte. 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	XN		High-speed alphameric character set for 1403, Model 2
YN	YN		High-speed preferred alphameric character set for 1403, Model 3 or N1

Figure 10. Special Character Sets for the 1403, 3203-5, and 3211 Printers (JES2)

Note: Where three values exist (for the 1403, 3211, and 3203-5 printers), either can be coded and JES2 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. In addition, the installation can design character sets to meet special needs and assign a unique code to them. See the system programming staff for a complete list of available character sets for the installation.

Requesting Character Arrangements with a 3800 Printing Subsystem

Character arrangement tables to be used when printing with the 3800 are specified with the CHARS parameter on the SYSOUT or output DD statement or on the JES2 OUTPUT statement. The table names supplied for the 3800 are given in the **IBM 3800 Printing Subsystem Programmer's Guide**. See your system programmer for the selection of table names available at your installation.

When more than one character arrangement table is specified, you can code OPTCD=J as a DCB subparameter to indicate that your data line contains a table reference character for dynamically selecting the table you want. (See the description of the OPTCD subparameter for BSAM and QSAM in the topic, "The DCB Parameter".) Using the IEBIMAGE utility program, you can modify or construct character arrangement tables and graphic character modification modules to allow substitution of existing or user-designed characters. Details for using both IEBIMAGE and the OPTCD subparameter are provided in the **IBM 3800 Printing Subsystem Programmer's Guide.**

The UCS (universal character set) parameter can be specified on the same output DD statement with the CHARS parameter to permit output to go to either the 3800 or to other printers. The GF10, GF12, GF15, or GFC character arrangement table coded on the CHARS parameter provides the same effect as the FOLD subparameter of UCS. If a printer other than the 3800 is selected for output, the CHARS parameter is ignored.

Requesting Forms Control

For a 1403 Printer and Printers Supported by Systems Network Architecture (SNA) Remote Job Processing (RJP): Forms control is requested by specifying a specific carriage control tape in the FCB parameter on the OUTPUT control statement. Carriage specifications are used for JES2 output processing only; they are ignored by the external writer.

For a 3211 or 3203-5 Printer: Specific forms control images (for example, the number of lines per page or number of characters per line) are requested by coding an image identifier in the FCB parameter on a DD statement or by coding FCB on the OUTPUT control statement. The FCB

two standard FCB images: STD1 and STD2. STD1 specifies that 6 lines per inch are to be printed on an 8.5-inch form. STD2 specifies that 6 lines per inch are to be printed on an 11-inch form. (Do not specify STD1 or STD2 for JES2 processing unless instructed by your installation.) Additional FCB images can be specified by the installation. For information on IBM- and user-supplied FCB images, see **OS/VS2 System Programming Library: Data Management.**

Note: JES2 treats the 3203-5 printer the same as a 3211 printer with the following exceptions:

- The UCSs for the 3203-5 are the same as for the 1403 printer.
- The 3203-5 printer does not support indexing; therefore JES2 indexing commands are ignored.
- You can not explicity identify the 3203-5 printer to JES2 via JES2 initialization parameters (MVS passes the 3203-5 identification to JES2 via the UCB).
  For further information on UCSs and UCBs, see **OS/VS2 System Programming Library: Data Management.**

For a 3800 Printing Subsystem: Forms control is requested by specifying an FCB module name in the FCB parameter on a DD statement or on the OUTPUT control statement. (Although the FCB image for the 3211 and the FCB module for the 3800 serve the same purpose, they are constructed differently and are not interchangeable between the two printers.) The FCB module provides a standard FCB module, STD3, which specifies output of 80 lines per page at 8 lines per inch on 11-inch long paper. (For a 3800 using ISO paper sizes, STD3 can be redefined by the installation.) Additional FCB modules can be specified by the installation. For information on IBM- and user-supplied FCB modules, see the **IBM 3800 Printing Subsystem Programmer's Guide.**

Requesting Forms Overlay

The forms overlay feature of the 3800 Printing Subsystem allows printing of the image from a forms overlay negative together with the data being printed. This reduces the need for pre-printed forms, and for changing of forms. The FLASH parameter on the DD statement or the FLASH and FLASHC parameters on the JES2 OUTPUT statement identify the overlay to be used and the number of copies on which that overlay is to be printed. When you do not specify FLASH on either the DD or /*OUTPUT statements the 3800 Printing Subsystem uses the default specified at JES2 initialization. For information on designing and making or obtaining forms overlay negatives, see the **Forms Design Reference Guide for the IBM 3800 Printing Subsystem.**

Bursting of Output

The optional Burster-Trimmer-Stacker of the 3800 Printing Subsystem separates continuous form paper into individual sheets. The BURST parameter on the DD statement or on the JES2 OUTPUT statement is used to specify to the operator whether the output is to go to the Burster-Trimmer-Stacker or to the continuous forms stacker. For further information and examples, see the topic, "The BURST Parameter".

your job to an output class (with the MSGCLASS parameter). At remote stations, JES2 offers most of the same options for writing data sets that are requested when submitting the job at the central computing center. You can request:

- That a data set be held until the operator requests that it be printed.
- That a special output form be used by specifying a form name in the SYSOUT parameter.
- That multiple copies of the data set be used.

Whether at a remote station or at the central computing center, you can also request that a data set be routed to another destination. To route an output data set to another destination,

Controlling Output Destination (JES2)

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

Unless overridden by the system programmer or operator, the default output location is the submitting location, either a remote work station or the central site (destination of LOCAL). To receive the output at the submitting location, simply assign output data sets to any output class (with the SYSOUT parameter) and messages from your job to an output class (with the MSGCLASS parameter). At remote stations, JES2 offers most of the same options for writing data sets that are requested when submitting the job at the central computing center. You can request:

- That a data set be held until the operator requests that is be printed.
- That a special output form be used by specifying a form name in the SYSOUT parameter.
- That multiple copies of the data set be used.

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

- /*ROUTE Statement (with PRINT and PUNCH options) - Allows the user to specify the destination of jobs or output for any node or any remote station. All output that has no other specific assignment is directed to the destination specified in the ROUTE statement.
- DD SYSOUT Statement - May be coded for a specific data set assignment by coding the DEST parameter on the DD SYSOUT statement, thus routing that data set to a particular destination.
- SYSOUT Parameter - Allows the user to specify a code name that points to a /*OUTPUT statement, which in turn contains a DEST parameter identifying the actual destination. This method allows the user to send more than one (but not necessarily all) data sets to the same place without coding DEST on each SYSOUT statement. If the destination should change, only the OUTPUT statement need be altered.
- Default Output Destination - Defined implicitly when the job enters the job entry network; the default destination is determined by the device upon which the job entered the system.

If you code a destination on either the SYSOUT or the OUTPUT statements, it will override the destination in the ROUTE statement. Work stations are identified by a destination identification that has been established by the system programmer. The destination parameter will cause output to be routed to local printers or punches or to any remote station.

Example of Obtaining Output

This example shows the use of JES2 and JCL statements that can be used to obtain output.

```
/*PRIORITY  5
//OUTJOB    JOB     BAKER,PERFORM=100,MSGCLASS=J
/*JOBPARM   COPIES=2,LINECT=20,ROOM=233,FORMS=GRN1
/*OUTPUT    PSET DEST=PRINTER8,FCB=STD3,FORMS=2PRT,UCS=TN
/*SETUP     SCHLIB
//STEP1     EXEC    PGM=TESTSYSO
//DD1       DD      DSN=DATA,UNIT=2314,VOL=SER=SCHLIB,
//                  DISP=(OLD,KEEP),SPACE=(TRK,(5,2))
//DD2       DD      DSN=&&TEMP,UNIT=2314,DISP=(NEW,DELETE),
//                  SPACE=(TRK,(10,5))
//DD3       DD      SYSOUT=(A,,PSET)
//DD4       DD      SYSOUT=(A,,GRPH)
//DD5       DD      SYSOUT=L
```

1. The job will be selected at priority level 5.
2. The job will run in performance group 100; the meaning of 100 is defined by the installation. All system messages are to be written to output class J.
3. The JOBPARM statement indicates that:
   a. Two copies of the entire job related output will be printed.
   b. No more than 20 lines per page will be printed (LINECT=20).
   c. The programmer's office number is 233. This appears on the separator page and is used for distributing output.
   d. Forms name GRN1 is the name of the form to be used by all data sets unless a specific form is defined on a DD statement.
4. The OUTPUT statement indicates that:
   a. PSET is the code that, when indicated on a SYSOUT DD statement, causes all parameters on the OUTPUT statement to override default parameters.
   b. The destination for the output is a printer and is number 8 if it is local print/punch routing; otherwise, PRINTER8 is equivalent to LOCAL.
   c. If the printer has the forms control buffer feature, STD3 defines the special forms control buffer image to be used for processing the job.
   d. Forms name 2PRT is the name of the forms for data sets coding PSET in the SYSOUT parameter.
   e. TN means text printing on a 1403 printer.
5. The SETUP statement indicates that volume SCHLIB should be mounted before this job begins processing.
6. SYSOUT data sets and message class are printed on the form called GRN1 except DD3 and DD4. The DD4 SYSOUT data set is printed on the form called GRPH; the DD3 SYSOUT data set is printed on the form called 2PRT, since the code name subparameter of DD3 contains the value PSET (referring to the /*OUTPUT statement).

Routing a Job Through the System (JES3)

JES3 reads a job into the system, satisfies the requirements requested on JCL and JES3 statements, optionally preschedules JES3 supported devices, selects the job for execution, and controls the processing of SYSOUT data sets. JES3 automatically performs most of these services. JES3 networking allows you to transmit selected jobs and system output (SYSOUT) data sets to another system in the network that can be a JES2 system, a VM/370 system, or another JES3 system. Using JES3 networking, jobs can execute at the originating node (a node comprises a JES3 global processor and all of its local processors) or at another node within the network. By coding JCL and JES3 parameters you can influence how these services are performed. For example, JES3 schedules a job for execution but you can customize how and when the job is processed by coding one or more of the parameters on the JES3 MAIN statement. The MAIN SYSTEM parameter is used to direct a job to a specific processor in a loosely-coupled multiprocessing environment and the MAIN SETUP parameter is used to modify JES3 device selection algorithms. In a loosely-coupled multiprocessing system, processors are connected by channel-to-channel adapters that pass information to the JES3 system operating on each processor. Shared direct access devices, which are read by the global and local processors, contain JES3 and system control blocks, and user data. One processor, the global, controls job selection for all processors in the system. Other processors are called local processors.

To ensure that one job is executed before another, use dependent job control (DJC). To attempt job scheduling by a certain time, use deadline scheduling (MAIN DEADLINE). To hold a job for any other purpose, such as availability of input data, code TYPRUN=HOLD on the JOB statement or use the HOLD parameter on the MAIN statement.

MVS includes support for controlling the processing rate of jobs and job steps. The installation defines a certain number of performance groups. Each of these defines a particular processing rate to be used for associated jobs or job steps. To associate a job or job step with an MVS performance group, code the PERFORM parameter on either the JOB or EXEC statements.

This section contains ten topics:

- Scheduling a Job
- Selecting a Processor
- Routing a Job in a Network
- Allocating Data Resources
- Selecting a Job
- Passing Information to the Job in Execution
- Restarting a Job
- Spool Partitioning
- Example of Routing a Job Through the System (JES3)
- Example of Routing a Job Through a Network (JES3)

Scheduling a Job

JES3 controls the selection of jobs for processing. When a job is read into the system, it is initially placed on a spooling volume. The JCL and JES3 statements are checked for syntax errors. If no errors are present, JES3 determines allocation requirements for the job. JES3 device selection takes place next. Devices are selected based on device requirements for JES3-managed devices established in the JCL. Any necessary volumes that require mounting are requested. (More information on this subject is found in the Allocating Devices section.) Once all JES3-managed devices are selected and the first volume on each device is mounted (unless

deferred mounting is requested or implicit high watermark setup is used), the job is placed in the queue for execution. (Implicit high watermark allocates a minimum number of devices to run a job.)

When JCL or JES3 statements have syntax errors, appropriate error messages are issued and the job is terminated. When the job has JES3 allocation errors, error messages are issued and execution is bypassed.

The execution queue is logically divided into groups of job classes specified by the installation and within each job class group, jobs are placed according to their job priority. Jobs in the same job class group with the same job priority are placed in the execution queue in the order they were read into the system. The various job class groups are assigned priorities by the installation. JES3 starts system initiators on each processor and assigns them a job class group to process based on the installation priorities. It selects jobs from any class assigned to it. Jobs are selected by job class, processor eligibility, workload balancing, and priority order as described in the section, "Selecting a Job".

Selecting a Processor

JES3 usually automatically selects a processor for a job based on device, volume, and data set dependencies known to it. However, if any of the dependencies are not known to JES3, the job can be processed incorrectly or can fail. The next section, "Allocating Devices," discusses these dependencies in more detail. There can also be processor dependencies; that is, a special system feature such as an emulator, non-standard catalog, or system-managed device, that JES3 will not recognize unless you define which processor is required using the SYSTEM parameter on the MAIN statement.

The MAIN SYSTEM subparameters, JGLOBAL and JLOCAL, request the global or a local MVS processor. To specify particular processors or exclude particular processors, code the main-name value on the MAIN SYSTEM parameter for each processor.

Not all classes are eligible to run on all processors; therefore, make sure that the job class for the job is eligible before selecting a specific processor.

A job is flushed if it specifies a job class (on the JOB or MAIN statements) and a specific processor(s) (on the SYSTEM and TYPE subparameters on the MAIN statement) that are incompatible. A processor(s) is defined for each valid job class on the CLASS initialization statement during JES3 initialization. For example, if a job specifies CLASS=C and SYSTEM=SY1, then the processor SY1 must have been defined on the CLASS initialization statement for class C.

If you do not specify the SYSTEM parameter, or if you omit the MAIN statement, the job is eligible to run on those processors for which its class is eligible.

If any DD statement in the job contains a device address in the UNIT parameter and that device is either JES3-managed or jointly-managed, you must use the SYSTEM or TYPE parameters to restrict job eligibility to the processor that has a path to that device.

Routing a Job in a Network (JES3 Networking)

To prepare a job for execution at a remote node:

1. Code a JOB statement the same as the JOB statement you normally code to execute a job at your installation.
2. Code a JES3 //*ROUTE XEQ statement. This statement specifies the node that is to execute the job. It causes JES3 to divert the incoming job stream to a special input service function. This special input service function transmits the data that follows the //*ROUTE XEQ statement to the specified node.
   - Without the //*ROUTE XEQ statement, the job will execute at the submitting node.

3. Code a JOB statement that allows the remote job to execute at the node specified in the //*ROUTE XEQ statement. Because procedures vary between installations the format of this JOB statement might be different from the format you normally use at your installation. For example, the accounting field of this JOB statement might or might not contain the same parameters in the same format that you are accustomed to using.

   - An error on the //*ROUTE XEQ statement might cause the second JOB statement to signal the start of a job for local execution. To avoid this possibility, specify NJB in the second JOB statement in place of JOB. NJB is changed to JOB by JES3 input service prior to transmitting the job through the network.

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

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

Consult the system programmers at your installation and at the remote installation for the specific information required to properly prepare jobs for execution at the remote node.

Factors to be considered are:

- The content and format of the JOB statement: The remote node might have different parameter requirements than the submitting node.
- The job entry subsystem in use at the remote node: The remote node, if using a different job entry subsystem, will have different control statement requirements.
- The content of the procedure library (PROCLIB): The procedure library might not be the same at the remote node as it is at the submitting node.
- Data set identification: To use data sets at the remote node you need to know the names and values to place in your JCL parameters to enable your job to gain access to these data sets.
- Esoteric unit names in use at the remote node might include devices different from the submitting node.
- SYSOUT classes might have different meanings at the remote node than at the submitting node.

Allocating Data Resources

Data resources, that is, devices, data sets, and volumes that are required for each DD request, are allocated either by JES3 or by the system according to DSNAME, DISP, UNIT, and VOLUME on the DD statement. If your data set is an existing data set and specifies or requires a unit managed by JES3, JES3 will allocate the request on a job basis before the job executes by examining the request in relation to other data requests in this and other jobs. Otherwise, the system will allocate the request on a step basis as the step enters execution.

When a job is read into the system, it is initially placed on a spool volume or volumes.

To prevent JES3 from spreading a job's spool data sets across all spool volumes, an installation can divide spool volumes into separate groups known as partitions. Spool partitioning allows your installation to assign job-related data to a particular spool partition by type or class of information. Depending upon how your installation defines its partitions, this can result in spool data for a particular job (or all information of a particular type - input, output, etc.) being allocated to the spool volume or volumes defined to a specified spool partition.

Devices are divided into three catagories in a JES3-controlled system: system-managed, jointly managed, or JES3-managed. The following chart indicates what type of devices are eligible for each type of allocation.

How Managed	Attribute of Device	
	Permanently Resident	Removable
MVS-managed only	X	X
Jointly managed (JES3/MVS)	X	
JES3-managed only		X

JES3 allocation (job setup, high watermark setup, and explicit setup explained later in this topic) will utilize JES3-managed devices and jointly managed devices. MVS allocation will utilize MVS-managed and jointly managed devices. The system programmer defines the devices that are in each category managed either jointly or by JES3. Refer to "Requesting Units and Volumes" earlier in this book for a brief discussion of system allocation. Refer to **OS/VS2 System Programming Library: Job Management** for additional information on system allocation, and **JES3 System Programming Library: Installation Planning and Tuning** for more information on JES3.

Spool Partitioning

At initialization, spool partitions are defined by the JES3 SPART initialization statement. A spool partition can be a single spool data set or a set of spool data sets. One of the partitions must be the spool partition that JES3 uses to allocate spool space to requests that do not have a spool partition assigned. If your installation does not specify the default spool partition explicitly, the default spool partition is the first spool partition defined by a JES3 SPART initialization statement. If your installation does not specify any spool partitioning at JES3 initialization, JES3 assigns all spool volumes to a JES3 supplied default spool partition (JES3PART) and all functions will use this default partition.

Output spool data sets are assigned to a specific spool partition by the JES3 SYSOUT initialization statement. Particular classes of jobs are assigned to a specific spool partition by the JES3 CLASS initialization statement. All jobs associated with a particular processor are assigned to a specific spool partition by the JES3 MAINPROC initialization statement.

Installation programmers can assign a specific job to a specific spool partition by coding the SPART parameter of the JES3 //*MAIN statement.

Note: Before using the SPART parameter on the //*MAIN statement, installation programmers should have a complete understanding of how JES3 allocates a job's data to spool partitions. This information is found in **JES3 System Programming Library: Installation Planning and Tuning.**

When JES3 assigns spool data sets to a spool partition, the spool data sets are assigned using the following priorities:

- The spool partition requested on the job's //*MAIN statement
- The spool partition associated with the job's output class and defined by a JES3 SYSOUT initialization statement
  Note: For output data sets, the spool partition defined by a JES3 SYSOUT initialization statement overrides all other spool partitions specified.
- The spool partition associated with the job's class and defined by a JES3 CLASS initialization statement
- The spool partition associated with the executing processor and defined by a JES3 MAINPROC statement
- The default spool partition

Code the SPART parameter on the //*MAIN statement as SPART=partition-name, where the partition-name is the spool partition that JES3 is to allocate for this job's input and output spool data sets. If a spool partition is defined with a JES3 SYSOUT initialization statement that is associated with this job's class then the SPART parameter on the //*MAIN statement is ignored for this job's output data sets. The partition name must match exactly the name of a spool partition coded in the NAME parameter of a JES3 SPART initialization statement.

If you incorrectly code the partition name on the //*MAIN statement, JES3 issues a warning message and assigns the job to the job class spool partition if one is defined. Otherwise JES3 assigns the job to the default spool partition.

If JES3 encounters more than one SPART parameter on a //*MAIN statement or statements, it issues a warning message and uses the last SPART value encountered.

When specifying a spool partition with the SPART parameter on a //*MAIN statement, the //*MAIN statement must precede the job's first EXEC statement and any JES3 //*DATASET or //*PROCESS statements in the job's input stream.

If JES3 input service reads an EXEC, //*DATASET, or //*PROCESS statement prior to a //*MAIN statement that defines a spool partition, JES3 ignores the SPART parameter on the //*MAIN statement and assigns the job to the default spool partition.

Refer to the following JES3 initialization statements when reading the examples that illustrate the use of spool partitioning.

```
*JES3 Initialization Statements
SPART,NAME=PARTA,DEF=YES
SPART,NAME=PARTB
SPART,NAME=PARTC
SPART,NAME=PARTD
SPART,NAME=PARTE
*
CLASS,NAME=BATCH,DEF=YES
CLASS,NAME=IMSBATCH,SPART=PARTB
*
MAINPROC,NAME=SY1...
MAINPROC,NAME=SY2,SPART=PARTC
```

```
*
SYSOUT,CLASS=N
SYSOUT,CLASS=S,SPART=PARTD,TYPE=(PRINT,DSISO)
```

The following examples illustrate the use of the SPART parameter on a //*MAIN statement.

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

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

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

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

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

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

Types of JES3 Setup

JES3 supports three types of setup: job setup, high watermark setup, and explicit setup.

Job setup. Job setup results in allocation of all JES3-managed units required in the job. If you specify SETUP=JOB on the MAIN statement, JES3 will mount the initial volumes necessary to run all steps before the job executes. When volumes are no longer needed, they will be demounted and the devices deallocated (made available for use by another job). If the

FREE=CLOSE DD parameter is specified, the deallocation takes place when the data set is closed.

High watermark setup. This setup is used to reduce the number of devices that are reserved for a job. With this setup, JES3 reserves only the maximum number of devices of each type needed during any one job step. High watermark setup does not cause premounting of all mountable volumes.

When high watermark setup is used, as in job setup, devices, volumes, and data sets are returned to JES3 for use by other jobs as soon as the DD statement is deallocated in the last step using the resources. When it is advantageous to use fewer devices for a job, high watermark setup is preferable to job setup.

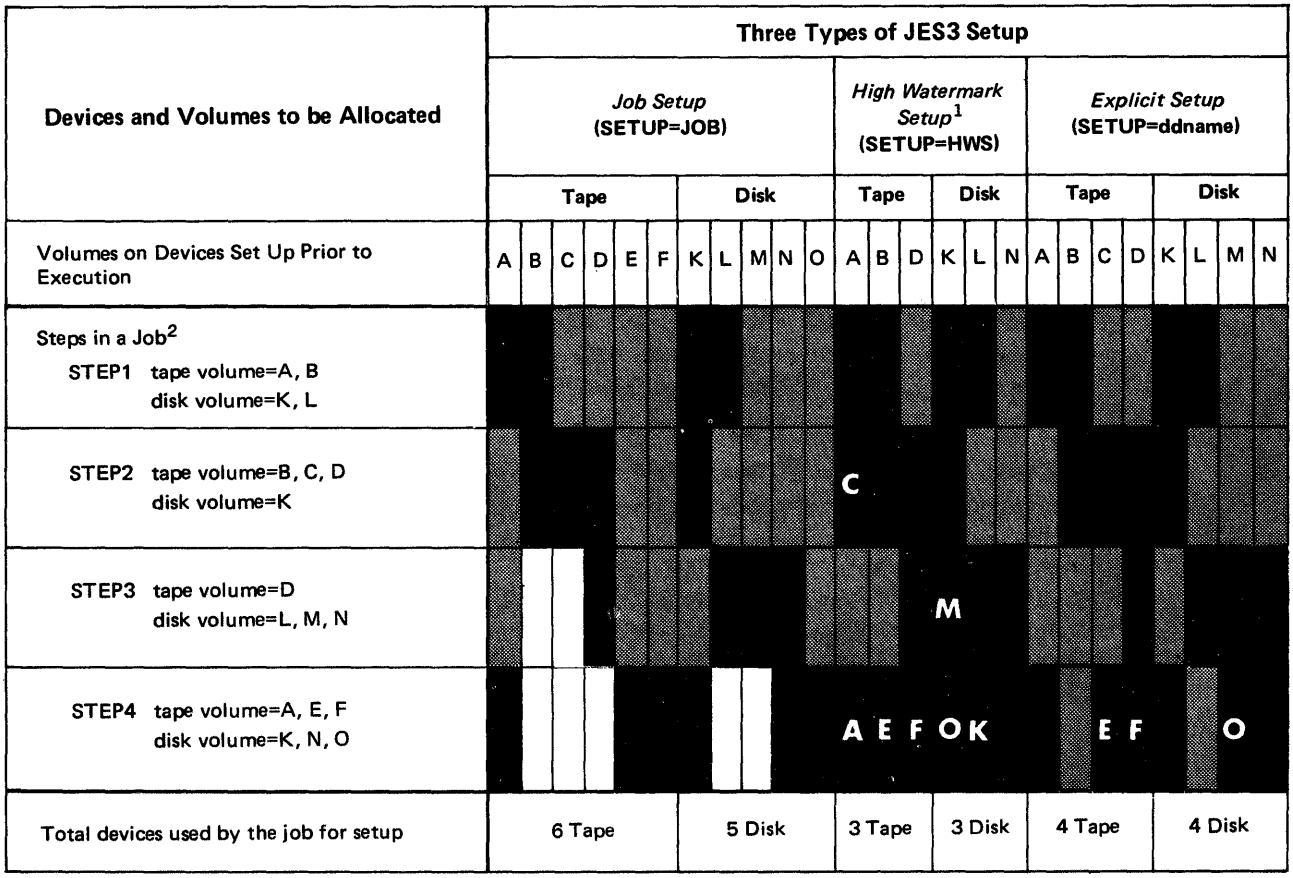
High watermark setup is used when you specify SETUP=THWS (for tapes only), SETUP=DHWS (for direct access devices only), SETUP=HWS (for tape, direct access devices, graphics, and unit record devices), or MSS=HWS (for MSS devices) on a //*MAIN statement for the job. If the SETUP or MSS parameter is not specified on a //*MAIN statement, high watermark setup is used only when SETUP=THWS, SETUP=DHWS, or SETUP=HWS is specified

on the STANDARDS initialization statement, or MSS=HWS is specified on the SETPARM initialization statement.

In Figure 11, volume A is mounted for use in STEP1 and then demounted when not in use until needed in STEP4. Volume K is mounted for use in STEP1 and STEP2 and then demounted when not in use until needed in STEP4. Volumes A and K are mounted in STEP4 on any available device.

Explicit setup. Explicit setup is user directed. It uses the number of units required by job setup, but premounts volumes according to the explicit setup specifications. SETUP=ddname or SETUP=/ddname on the MAIN statement specifies explicit setup and the ddname request to be setup or to be removed from consideration for setup. An advantage of explicit setup over high watermark setup is that volumes can be forced to remain mounted on devices until they are no longer needed. However, there is one disadvantage if explicit setup is specified: there is no early deallocation of devices. Job setup and high watermark setup can deallocate devices at the end of any step if the devices are no longer needed. Explicit setup, however, allocates a certain number of devices before job execution and does not deallocate any until the job completes execution.

In Figure 11, four devices are allocated for both tape and disk instead of the three allocated using high watermark setup. By explicitly requesting that certain volumes be mounted, volumes A and K can avoid being deallocated and remounted for the last step.



LEGEND:

The device is allocated and in use

The device is allocated but not in use

The device is no longer needed and can be deallocated

[1]High watermark setup can express combinations of tape and disk allocations.
HWS request allocation of the minimal number of devices required to run the job.
THWS requests high watermark setup for tapes and job setup for disks.
DHWS requests high watermark setup for disks and job setup for tapes.

[2]Volumes mounted after STEP1 are indicated by placing the volume name in the box for the step in which it is allocated. For example, in high watermark setup, volume C is mounted at STEP2.

Figure 11. Example of JES3 Setup

To avoid holding devices until the step needs them for any of the forms of JES3 allocation, use dependent job control. See the topic, "Dependent Job Control," later in section to determine how to split a job into smaller, dependent jobs for execution.

Selecting a Job

Jobs are selected for execution according to processor eligibility, job class, job processing balancing, and priority order. A job must first be eligible for a particular processor, then selection is by class (as defined by installation criteria) and optionally by workload characteristics and by priority. Processor eligibility is discussed in the section, "Selecting a Processor."

Assigning a job to a job class. A job class is a description of the type of job being submitted; that is, production, testing, and so forth. It is established by the installation and has no inherent meaning except as the installation has defined it. It is used by the installation for scheduling jobs on eligible processors. To assign a job to a job class, code the CLASS parameter on the JOB statement or the CLASS parameter on the MAIN statement. If neither of these parameters are coded, the job will be assigned an installation-defined standard class default.

Establishing job processing balance. The MAIN IORATE parameter specifies a value for the job to determine the mix of jobs for each processor. It defines the relationship between CPU-bound processing and I/O-bound processing for that job which is expressed as being high, medium, or lowI/O. JES3 attempts to provide a balance of CPU-bound and I/O-bound jobs to improve the scheduling of jobs for execution. The MAIN IORATE parameter regulates how a job is scheduled as contrasted with the PERFORM parameter on either the JOB or EXEC statement that regulates how a job executes. The PERFORM parameter is discussed in the section, "Performance of Jobs and Job Steps."

Assigning a priority to a job. Within a job class group, jobs are selected for execution according to job priority. Jobs with the same priority are placed in a first in/first out order. Specify job priority by coding the PRTY parameter on the JOB statement.

The priority order for jobs can be changed by the operator, by priority aging, or by deadline scheduling. How the operator changes priority is discussed in **JES3 Operator's Library**.

Priority aging allows JES3 to increase the priority of a job after it has been passed over by JES3 an installation-specified number of times. A job can be bypassed because of an insufficient number of devices or contention for a volume or data set or because there is not enough main storage on an MVS processor. The installation defines priority aging; it cannot be specified using JCL.

Deadline scheduling allows you to specify a time of day when the job should be scheduled. If the job is not scheduled by this time, JES3 will increase the priority of the job at installation-defined intervals until it is scheduled. For more information on deadline scheduling, refer to the next section.

In addition to job selection, raising a job's priority will cause the job to be given preferential treatment in JES3 device selection. For more information on JES3 device selection, see "Allocating Devices."

Deadline Scheduling

When a job must be scheduled by a certain time of the day, week, month, or year, specify this on the MAIN DEADLINE parameter. By indicating that there are time restrictions, you influence the priority of the job and help insure that the job will be scheduled when necessary. For example, a job must be scheduled every Friday by 2 p.m. to calculate the payroll. Request that the job be scheduled by that time by coding //*MAIN DEADLINE=(1400,A,6,WEEKLY). The subparameter values mean the following:

1400 is 2 p.m. on a 24-hour clock.
A defines the deadline type that determines the periodic increment of the job's priority (the meaning for A is defined by the installation).
6 is the sixth day of the week (the first day is Sunday; the seventh day is Saturday).
WEEKLY is the cycle indicating the frequency of scheduling this job.

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

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

Postponing Job Selection

It is possible that resources other than those managed by JES3 will not be available; for example, you may want to read a job before all input is available. In this case, delay the job's selection by coding TYPRUN=HOLD on the JOB statement or HOLD=YES on the MAIN statement. When delaying a job's initiation, the job remains on the selection queue but cannot be selected for processing until the operator releases the job. Notify the operator when to release the job on the JES3 OPERATOR statement. When the operator releases the job, it is again eligible for selection.

Performance of Jobs and Job Steps

To regulate the execution performance of a job, associate a job or job step with a performance group. The installation defines performance groups that determine the rate at which a given job will have access to the CPU, storage, and I/O channels. Most performance groups designate good processing rates under light system workload conditions. However, when the system workload is moderate or heavy, some performance groups will have significantly lower processing rates than others. The installation defines the performance groups needed to meet the response requirements of its various users and will probably publish this information for your use. Associate the performance group with a job or job step by coding a performance group number on the PERFORM parameter on the JOB or EXEC statements. The PERFORM parameter regulates how a job executes as contrasted with the MAIN IORATE parameter that regulates how a job is scheduled. The MAIN IORATE parameter is described in the section, "Selecting a Job."

For further information concerning system performance, refer to **OS/VS2 System Programming Library: Initialization and Tuning Guide.**

Assigning a Dispatching Priority to Job Steps

For most jobs run on MVS, the job's dispatching priority will default to an automatic priority group (APG) instead of being assigned. The automatic priority group function is an algorithm that the system resources manager will use to attempt to increase system throughput by dynamically adjusting the dispatching priority of associated address spaces.

If you do assign a dispatching priority, code the DPRTY parameter on the EXEC statement. This parameter has two values that the system uses in the following formula to calculate the dispatching priority:

```
(value1 x 16) + value2 = step's dispatching priority
```

If you omit the DPRTY parameter completely, the job step is assigned the APG priority. If value1 is omitted or it is equal to the APG value, the step is assigned the APG priority and any value you code for value2 is ignored. In this case, value2 is obtained from the Installation Performance Specification (IPS) using the performance group associated with the job step. (See **OS/VS2 System Programming Library: Initialization and Tuning Guide** for information on IPS.) If value2 is not specified in the IPS, a value of 6 is assigned to value2.

Dependent Job Control

Dependent job control (DJC) is used when jobs must be executed in a specific order determined by job dependencies. There are several reasons for requiring one job to process before another. For example, in JES3, data set information is fetched from the catalog before the job is scheduled. If JOBA changes or adds to a catalog that JOBB will refer to, use dependent job control to ensure that JOBA runs before JOBB is processed by JES3 allocation. Another reason for using DJC is to achieve better device utilization. If a job requires only one device for the first four steps but requires five devices for the fifth step, break the job into two jobs (one for the first four steps and one for the fifth step). Use DJC to make the second job dependent on the first; that is, the second job can run only after the first job has completed. DJC is also useful in controlling the scheduling of jobs that have data dependencies.

To define a dependent job net, submit a NET statement with each job. The NET statement identifies a job's net and specifies the dependency that must be satisfied before the job can be scheduled. Jobs normally must wait for scheduling until a predecessor job completes. Jobs that depend on one or more predecessor jobs to complete are called successor jobs. To specify the number of predecessor/successor relationships of a given job in a net, specify the number of predecessor jobs on the NHOLD parameter and the name of each successor job in the RELEASE parameter of the NET statement. The number of predecessors is the number of jobs immediately prior to the job dependent upon other jobs completing; the number of successors is the total number of all jobs remaining to be processed in the net that depend on this job completing.

A normal or abnormal predecessor completion can be the requirement established for going to the next job. For example, JOBAB might not be requested unless the predecessor job abnormally terminates. The NORMAL and ABNORMAL parameters on the NET statement

specify the kind of predecessor completion required for the successor dependent job to execute.

A job-network job that has previously completed normally can be resubmitted while the associated network is still active in the system. The resubmitted job does not enter the network nor does it affect the network's processing. If you want to resubmit the previously completed job as part of the network, you must first free the network from the system and then resubmit the entire network.

The number of immediate predecessor jobs that must complete before a job is released for scheduling, including jobs from another network that are predecessors to the dependent job, are specified on the NHOLD parameter. When this parameter is defined, the job is placed into dependent job control hold status when it enters the system. A job is made eligible for JES3 allocation and scheduling when its NHOLD count, which is decremented when each predecessor job completes execution or by the operator, becomes zero. However, the NHOLD value can be decremented before the predecessor job completes execution by issuing a DJC (dependent job control) WTO (write-to-operator) message in the predecessor job problem program. Refer to **OS/VS2 Supervisor Services and Macro Instructions**, for the format of the write-to-operator command and **JES3 System Programming Library: Installation Planning and Tuning** for message text information.

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

Upon either normal or abnormal completion of a predecessor job, a successor job can have its NHOLD count decremented, can be flushed from the system, or can be retained pending operator action. If it is flushed, the job and all of its successor jobs (and their successor jobs) are canceled, printed, and flushed from the system. If it is retained in the system in the held state, the NHOLD count is not decremented and the job and all of its successor jobs are suspended from scheduling until either the predecessor is resubmitted or the operator decrements the NHOLD. You can control external dependencies by setting the NHOLD value one greater than normally assigned and asking the operator to decrement the NHOLD count when the dependency is satisfied.

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

Dependencies can be established between jobs in different nets. To indicate that a job in one network is the predecessor to a job in another network, specify the NETREL parameter. See the of dependent job control example on the following pages to show the use of the NETREL parameter.

Devices can be dedicated to a dependent job net by coding the DEVPOOL parameter. When the DEVPOOL parameter is coded in the first job in a net, (it is ignored if not coded in the first job) the devices specified are dedicated for device allocation and volume mounting only by jobs in the same net. To release these devices prior to all jobs completing in the network, code the DEVRELSE parameter. This parameter may be specified on one or more jobs in the net, except the first job. The first completing job that contains DEVRELSE=YES will cause the dedicated devices to be released. If no such job is encountered, the devices are released when the net is purged.

How to Code NET Statements

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

1. Draw a diagram of the net, connecting dependent jobs with lines indicating the flow of job dependencies. Give the net a name (such as EXAM1) to identify the net; this becomes the NETID parameter value.
2. List the jobname of each job in the net in the order of their dependencies on one another. Note next to each jobname the number of predecessors to the job, including predecessors of other job-nets, if applicable. The number of predecessors becomes the NHOLD parameter value. If early setup scheduling is desired, specify it as RS=count (RELSCHCT=count) where count specifies setup of a dependent job's resources before all of its predecessors have completed execution.
3. List the disposition of each successor jobname based on normal or abnormal predecessor completion.
4. List the successor jobnames for each job in the diagram. If there is a successor in a different net, then list the successor jobname and successor net-id in parentheses. The successors become the RELEASE parameter values.
5. Construct the necessary NET statements based on the diagram.

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

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

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

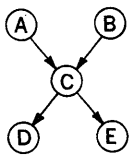
Examples of Dependent Job Control

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

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

1. A simple net

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



NETID EXAM1	Jobname	Predecessors (NHOLD)	Successors (RELEASE)
	A	0	job C
	B	0	job C
	C	2	jobs D,E
	D	1	none
	E	1	none

How to code EXAM1:

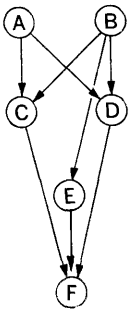
Jobname	Control	Statement
A	//*NET	NETID = EXAM1,RELEASE = (C)
B	//*NET	NETID = EXAM1,RELEASE = (C)
C	//*NET	NETID = EXAM1,NHOLD = 2,RELEASE = (D,E)
D	//*NET	NETID = EXAM1,NHOLD = 1
E	//*NET	NETID = EXAM1,NHOLD = 1

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

2. Multiple predecessor jobs

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

the NETID is EXAM2.



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

How to code EXAM2:

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

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

3. Complex network

Given: two networks, EXAM4 and EXAM3.

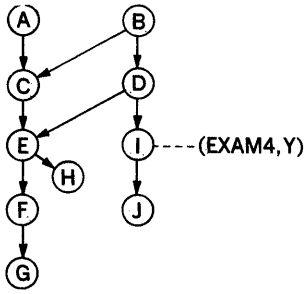
EXAM4 contains four jobs, W,X,Y, and Z.

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

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

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

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

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

(EXAM4,Y)

How to code EXAM4:

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

How to code EXAM3:

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

Conditional Execution of Job Steps

Depending on the results of one step of a job, you may not wish to execute subsequent steps — if a compilation fails, you would not want to waste computing time attempting subsequent link-editing or execution steps. Specify tests to determine whether to bypass or execute job steps based on the results from previous steps by coding the COND parameter on the JOB or EXEC statements.

The results of a job step can be reflected in a **return code**, a number from 0 to 4095. The COND parameter can be coded to test the return codes which are issued by the compiler, assembler, and linkage editor programs. Some return codes are standard for certain programs; for example, a return code of 8 issued by a compiler or linkage editor indicates that serious errors were found and execution is likely to fail. In problem programs, assign a number as the return code to signify a certain condition. For example, if STEP1 of a job reads accounts to be processed in subsequent job steps, set a return code of 10 if no delinquent accounts are found. Before STEP3 is executed to process delinquent accounts, test the return code from STEP1; if the return code from STEP1 is 10 — there are no delinquent accounts — skip STEP3. Specify the test to check the return code from STEP1 by coding the COND parameter.

You can also instruct the system to execute a step even if a previous step has abnormally terminated or only if a previous step has abnormally terminated by coding EVEN or ONLY in the COND parameter on a EXEC statement. For example, STEP1 of a job updates records in a data set. If STEP1 abnormally terminates, you want to execute STEP2, which will print the data set. Specify that STEP2 should be executed only if STEP1 abnormally terminates by coding ONLY in the COND parameter on the EXEC statement for STEP2.

Note: When JES3 determines the setup requirements for any given job, it does so without regard to any COND parameters that may be specified on the EXEC statements. All jobs are processed as though each step will be executed. This is necessary because setup requirements are determined in advance of job execution. The JES3 interpreter DSP (dynamic support program) has no way of predicting whether any given step will or will not execute, or what return code the program will provide.

Specifying Return Code Tests

In the COND parameter, specify tests to determine if the system should bypass a job step. If the system determines that a comparison is true, the job step is skipped (if COND was coded on the EXEC statement) or all remaining job steps are skipped (if COND was coded on the JOB statement). (Note that when a step is bypassed, return code tests are ignored.)

For example, COND=((10,GT),(20,LT)), asks "Is 10 greater than the return code or is 20 less than the return code?" If the return code is 12, neither test is satisfied: no job step is skipped. All the tests specified must be false if processing is to continue without skipping any job steps. If the return code is 25, the first test is still false, but the second test is satisfied: 20 is less than 25. The system will bypass one job step or all remaining job steps, depending on whether the COND parameter was coded on the EXEC statement or on the JOB statement.

Passing Information to the Job in Execution

Some information required by a program may vary from application to application, such as module attributes and options required by the compiler, assembler, and linkage editor programs. To provide this information to the program at the time it is executed, code the PARM parameter on the EXEC statement. The program must include instructions that can retrieve this information. (The exact location and format of the information passed to a processing program are included in **OS/VS2 Supervisor Services and Macro Instructions.**)

The PARM parameter can also be coded on the EXEC statement of a cataloged or in-stream procedure step. This establishes fields in which information is passed to the job. Override, add, or nullify parameters in a procedure or define symbolic parameters by coding the PARM parameter on the EXEC statement of the job calling a cataloged or in-stream procedure. For more information on the PARM parameter for these features, see "Cataloged and In-stream Procedures."

Identifying the Program to be Executed

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

Temporary Library

To assemble, link edit, and then execute in a single job, make the output of the linkage editor a member of a partitioned data set. This is accomplished by creating a temporary library; that is, a partitioned data set used to store a program until it is executed in a subsequent job step. When the program is required, refer back to the DD statement that defines the temporary library and the member by coding PGM=*.stepname.ddname or PGM=*.stepname.procstepname.ddname. Also request use of a program that is a member of a temporary library by coding PGM=program name and including a DD statement named JOBLIB or STEPLIB that defines the temporary library. To keep this program available for use by other jobs, make the program a member of a private library. For more information on temporary libraries, refer to the section, "Creating and Using Private and Temporary Libraries."

Private Library

To use a program that is a member of a private library, code PGM=program name and include a DD statement named JOBLIB or STEPLIB that defines the private library. The system automatically looks in the private library for a member with the corresponding name.

A program that resides in a private library can also be executed by coding PGM=*.stepname.ddname or PGM=*.stepname.procstepname.ddname. This can be done only when the named DD statement defines the program as a member of a private library. For more information on private libraries refer to the section, "Creating and Using Private and Temporary Libraries."

The IEFBR14 Program

This program is used to check the syntax of the control statements, allocate space, or satisfy requests for disposition processing prior to execution of a job. To do this, substitute IEFBR14 for the program name on the EXEC statement. When this program is called, it gives a return code of 0 and returns to the calling routine. For an example of the IEFBR14 program, see the example in the topic "How to Code NET Statement" in this section.

If you created a data set when using this program, the data set's status will be old when you execute your program. For situations when a disposition of CATLG will not be satisfied see "Cataloging a Data Set" in this publication.

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

Testing JCL Without Execution

page 151

There are methods other than IEFBR14 for testing JCL. The TYPRUN=SCAN parameter on the JOB statement or the PGM=JCLTEST and PGM=JSTTEST parameters on the EXEC statement cause the system to scan the JCL for syntax errors without processing the job or setting up devices. (For further information about JSTTEST, see **JES3 System Programming Library: Diagnosis**. These parameters will check for invalid keywords, illegal characters, parentheses errors, and excessive parameters.

Selecting a Cataloged Procedure Library

You can choose which of the installation specified cataloged procedure libraries will be used for resolving catalog procedure references in the JCL by coding the PROC parameter on the MAIN statement. If this parameter is omitted, the installation standard library, denoted by ST, will be used.

If you want to update a cataloged procedure library, whether or not that library was used to resolve the job's library references, code the UPDATE parameter on the MAIN statement pointing to the library to be updated by that job. JES3 will effectively disable the use of that library, placing all jobs that request it into a held state until the updating job terminates. This prevents the use of the library while the update occurs. If concatenated data sets are defined, the library update facility cannot be used for any of these data sets.

Reading Column Binary Input

Jobs that require input from column binary cards can receive this input directly from the DD statement if JES3 is used by coding the MODE=C DCB subparameter on the DD * or DD DATA statement that precedes the column binary card input and by notifying the operator to read this job into a card reader for which he has specified mode C processing.

The DATASET statement can also be used to read column binary input for installation-written routines executed as part of nonstandard jobs. See the section, "The PROCESS Statement" for a discussion of nonstandard jobs.

Restarting a Job

When a job step abnormally terminates, you may have to resubmit the job for execution; this means delay and lost computer time. The operating system provides restart facilities to reduce the effects of abnormal termination.

If you have installed MVS/System Product-JES3 Release 2 (5740-XYN) without installing Data Facility/Device Support Release 1 Enhancement (5740-AM7) do not use the checkpoint/restart facility.

If you have installed Data Facility/Device Support with MVS/System Product you can use the checkpoint/restart facility with certain restrictions.

For additional information concerning these restrictions see **OS/VS2 MVS Checkpoint/Restart** and **OS/VS2 MVS System Programming Library: Job Management.**

There are three types of restarts:

- **Step restart**, from the beginning of a job step.
- **Checkpoint restart**, from a checkpoint within a job step. Establish checkpoints in a job step by coding the CHKPT macro instruction for each checkpoint. The CHKPT macro is described in **OS/VS2 MVS Data Management Macro Instructions**. See also the DD CHKPT parameter. It specifies that checkpoints are to be taken at end of volume for the data set defined by the DD statement on which it is coded.
- **System failure restart**, by specifying the FAILURE=RESTART parameter on the JES3 MAIN statement. In the event the job cannot complete executing because of system failure, JES3 will automatically reschedule the job from the beginning. Other options on the FAILURE parameter are CANCEL, HOLD, and PRINT. All of the values are described on the MAIN FAILURE parameter in the section, "Coding JES3 Statements."

Whether using step restart or checkpoint restart, the restart facility can be automatic or deferred.

Automatic restart: To use automatic restart, code the RD (restart definition) parameter on the JOB or EXEC statement. JES3 creates a job journal for any job specifying the RD parameter. (A job journal is established to hold restart information for each program in execution.) When a system failure occurs or a job step abnormally terminates and you have a job journal, the restart facility allows automatic restart when RD=R is coded on the JOB or EXEC statements. If checkpoints are taken, you can automatically restart at the last checkpoint regardless of whether or not the RD parameter is coded. When a job step abnormally terminates or a system failure occurs while the job is in execution and the installation has not implemented job journaling, these jobs are ineligible for automatic restart.

Deferred restart: To use deferred restart, code the RESTART parameter on the JOB statement. This required parameter specifies a job step or a step of a cataloged procedure and can specify a checkpoint identifier if you are using deferred checkpoint restart. The effect of the parameter is simply to restart the job at the beginning of the specified step or checkpoint. The SYSCHK DD statement is required when a job is being submitted for deferred checkpoint restart and must be placed immediately after a JOBLIB DD statement.

Refer to **OS/VS2 MVS Checkpoint/Restart** for a complete description of planning for and using the checkpoint restart facility.

Example of Routing a Job Through the System (JES3)

```
//EXAM      JOB
//*MAIN     SYSTEM=(MAIN1,MAIN2),SETUP=(DD1,DD2),
//*FAILURE=RESTART
//STEP1     EXEC    PGM=IEFBR14
//DD1       DD      UNIT=3330,VOL=SER=VOL1,DISP=OLD,
//                  DSN=JES3.EXAM
//STEP2     EXEC    PGM=IEFBR14
//DD2       DD      UNIT=3330,SPACE=(TRK,1),VOL=SER=VOL2
```

1. This job is assigned to an installation defined job class default.
2. The selection for processing is based on the priority assigned to the default class.
3. The MAIN statement specifies that this job can be processed on either MAIN1 or MAIN2. The job requires that the volumes associated with the DD1 and DD2 statements must be mounted.

Example of Routing a Job Through a JES3 Network

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

//REMOTE is a valid JOB statement for the submitting JES3 location. The //*ROUTE XEQ statement directs the following job to the remote node MSC. //EXAM is a valid JOB statement that allows the job to execute at the remote location specified on the //*ROUTE XEQ statement (MSC). ***Note:*** TSO users submitting jobs through a JES3 network must specify NJB in place of JOB on the second JOB statement.

You can request output by coding JCL and JES3 control statements. By coding JCL statements, you can request output data sets, listings of JCL statements, system messages, and abnormal termination dumps. By coding JCL and JES3 statements, you can request special forms processing, routing of output to specific devices, and multiple original printing by data sets within a job. The JES3 statements have the same options as JCL with some additional features such as the FORMAT statement forms overflow, forms control, and multiple data set characteristics.

This section includes six topics:

- Requesting Listings of JCL Statements and System Messages
- Requesting an Abnormal Termination Dump
- Writing Output Data Sets
- Controlling Output Destination
- Remote Job Processing
- Example of Obtaining Output (JES3)

Requesting Listings of JCL Statements and System Messages

The system produces messages about a job concerning allocation of units and volumes, disposition of data sets, and termination of job steps and the job. Request that these messages and/or the JCL statements from the job and from cataloged procedures called by the job be included on an output listing.

By coding the MSGLEVEL parameter on the JOB statement, you inform the system of what statements and messages you want included on the output listing. (The notation used on the output listing to identify cataloged and in-stream procedure statements is described in the chapter "Using Cataloged and In-stream Procedures.")

By coding the MSGCLASS parameter on the JOB statement, you assign messages and JCL statements to an output class. A default is assigned if MSGCLASS is not coded.

The JES3 FORMAT statement allows you to specify the ddname of the DD statement that defines the output data set characteristics you want to specify. If you want system messages, code DDNAME=SYSMSG; if you want the jclfile including statement messages, code DDNAME=JESJCL; or if you want JES3 and system operator messages (job log), code DDNAME=JESMSG.

Requesting an Abnormal Termination Dump

To obtain a dump in the event of abnormal termination of a job step, code a DD statement defining a dump data set. The name of the DD statement must be either SYSABEND, SYSMDUMP, or SYSUDUMP. If more than one of these DD statements is present the system uses the last one. If you override a DD statement in a cataloged procedure the overriding DD statement is used because it is the last one encountered.

SYSABEND, SYSMDUMP, and SYSUDUMP DD statements can provide a dump containing the processing program's virtual storage areas, the system nucleus, the entire system queue area, all local system queue areas, and any active link pack area (LPA) modules for the failing task. If either the Generalized Trace Facility (GTF) or the System Trace is active, the dump will

contain their records. (Both cannot be active at the same time and GTF has priority.) In addition, if your installation permits dumping of the common storage area, a SYSMDUMP DD statement can provide a dump containing those parts of the CSA the failing program is authorized to reference.

You can take subsequent SYSMDUMPS to the same data set if the data set name is SYS1.SYSMDPxx (where xx can be 00 - FF) and the data set disposition is SHR. Before attempting to take subsequent SYSMDUMPs see "Rules for Coding" under "The SYSABEND, SYSMDUMP and SYSUDUMP Facilities" in this publication.

If your program has issued an ABEND macro, or if you have written a recovery routine, you can determine what dump options you want, in addition to the installation defaults, and define them in a dump option list on the ABEND macro or on a SETRP macro issued by your recovery routine. How to do this is explained in **OS/VS2 Supervisor Services and Macro Instructions.**

Dumps with more data per page are available with the 3800 Printing Subsystem. By specifying CHARS=DUMP on the related DD statement, the dump is formatted in a 204-character line containing 64 bytes of storage. If FCB=STD3 is specified, the page is printed at 8 lines per inch. The dump program recognizes only STD3 for producing 8 lines per inch.

Descriptions of dumps and information on reading dumps are included in the **OS/VS2 System Programming Library: Debugging Handbook and OS/VS2 System Programming Library: Diagnostic Techniques.**

To have the dump which was produced for a SYSABEND or SYSUDUMP DD statement printed, either assign the dump to an output class using the SYSOUT parameter on the DD statement, or code the UNIT parameter and specify the printer on which you want the dump printed. To store the dump, define the data set as you would any other data set, coding the DSNAME, DISP, UNIT, and VOLUME parameters. If the data set will go to a direct access device, code the SPACE parameter.

The dump taken for a SYSMDUMP DD statement is machine-readable (unformatted) and must be stored on either a magnetic tape unit or a direct access device. If the job or step is running with nonpageable virtual storage (ADDRSPC=REAL on the JOB or EXEC statements), the SYSMDUMP output must be directed to a VIO data set.

To format and print a dump taken for a SYSMDUMP DD statement, use the AMDPRDMP service aid which is documented in **OS/VS2 System Programming Library: Service Aids**, or IPCS which is documented in **OS/VS2 MVS Interactive Problem Control System (IPCS) System Information.** Do not print the dump by sending it to SYSOUT=A because the output will be unformatted and difficult to read.

If you are using IPCS to format and print a dump taken for a SYSMDUMP DD statement, the data set disposition specified will produce the following results:

- DISP=MOD on a SYSMDUMP DD statement permits the collection of dump information pertaining to each of multiple events that occur during one job step.

Use of DISP=MOD produces a data set which can only be processed by IPCS (although IPCS's COPYDUMP subcommand can generate AMDPRDMP input from any dump collected in such a data set).

- DISP=NEW or DISP=OLD on a SYSMDUMP DD statement permits the collection of only the dump information pertaining to the last of multiple events which occur during one job step.

Use of either DISP=NEW or DISP=OLD produces a data set that can be processed by either IPCS or AMDPRDMP.

If you specify a private data set for SYSABEND or SYSUDUMP and more than one dump is possible, specify a disposition of MOD as it will be closed after each dump.

Writing Output Data Sets

There are two ways to write output data sets:

- Assign the data set to an output class.
- Specify the device on which the output should be written.

When you assign a data set to an output class, it is handled by JES3. The data set is first written to the JES3 spool device and then written to the final output device by either JES3 or an external writer. When you specify the device on the UNIT parameter, if the device is available, it is exclusively assigned to your job and under the control of your program.

Assigning Output Data Sets to Output Classes

Output classes include output with similar characteristics that are written to the same device. There are 36 possible output classes, defined by an alphabetic (A-Z) or numeric (0-9) character, that can be coded on either the SYSOUT or MSGCLASS parameters. The letter and number names have no inherent meaning; — each installation defines its own output classes and can assign special processing characteristics for each class. For example, output class W

might contain low priority output; class X might contain output to be printed on a special form (eliminating the need to request the form directly); class J might be reserved for high-volume output. If you want the output data set and messages from the job to be printed on the same output listing, specify SYSOUT=* or the same output class in the SYSOUT parameter as specified for messages in the MSGCLASS parameter.

The installation can designate certain SYSOUT data sets as reserved. Reserved classes can cause the data sets to be held; that is, not sent to JES3 output processing. If the output class specified for the MSGCLASS parameter is not designated as a reserved class, it will not be held and none of the job's data sets assigned to reserved classes will be held. Data sets can be explicitly held by coding the HOLD=YES parameter or by coding TSO commands. (Refer to **OS/VS2 TSO Command Language Reference** for information on the TSO commands.) Jobs are released from the hold state for the entire job by the operator or by the time-sharing user, with the OUTPUT TSO command. By using reserved classes, the controlling of the holding or not holding of all desired print data sets is done by means of the MSGCLASS parameter on the JOB statement.

Specifying the Device

To write an output data set without using JES3 output service, code the UNIT parameter on the DD statement defining the device on which the data set is written. The system will allocate the device exclusively to the job if the device is available; no other job can write output to that device until it is released. Jobs cannot share an output device as they can when the output is assigned to output classes.

Data management routines write the output from the program to the device specified in the UNIT parameter. Specifying a particular output device in the UNIT parameter generally is not the most efficient method for obtaining system output.

Processing Output Classes

Using JES3 is an efficient way to write output. JES3 supports the use of local and remote printers and punches as devices on which output classes are written. An external writer supports tape and direct access devices and user-written writer routines.

Output will be printed on the same listing if such parameters as CLASS, FORMS, FCB, UCS, and DEST have similar characteristics for all data sets and a user-written writer is not specified.

For an external writer, the operator will determine what data sets will be selected. When an external writer is specified, an IBM-supplied writer or a user-written writer will receive the output. The external writer must be started by the operator to have the data written to an output device. If you want to know more about how to write an external writer routine, refer to **OS/VS2 System Programming Library: Job Management.**

Output data sets to be written to a 3540 diskette must be assigned to an output class that is processed by the diskette writer (an external writer) as described in **OS/VS2 IBM 3540 Programmer's Reference.** For the diskette writer to receive data sets, the JES3 initialization deck must specify the SYSOUT classes to be reserved for diskette output. To write data sets on a diskette, the operator must start the diskette writer to a 3540 device.

Delaying the Writing of an Output Data Set

Data sets can be delayed from normal printing or delayed for inspection from a time sharing terminal prior to actually printing on that terminal by specifying reserved classes (as discussed next) and by coding the HOLD parameter. For example, the installation can direct the delayed

printing of a very large data set to prevent monopolizing an output device until smaller data sets are written. If a data set requires special forms that are not immediately available, it can be held until the operator supplies those forms. When HOLD=YES is specified on the DD statement, the data set is placed on a hold queue until the operator releases it. Notify the operator (using the OPERATOR statement for JES3) when that data set is ready for processing because no message will be sent to the operator. The data set must be released by the operator or time-sharing user for printing.

Suppressing the Writing of an Output Data Set

Whether writing an output data set by coding the SYSOUT parameter or the UNIT parameter, you can suppress the writing of the data set by defining it as a dummy data set. This is useful when testing a program and you do not want data sets printed until you are sure they will contain meaningful output. Suppressing the writing of a data set saves processing time.

If you are routing an output data set by coding the SYSOUT parameter, code the DUMMY parameter to define the data set as a dummy data set. When DUMMY is coded, the SYSOUT parameter is ignored and the data set is not written.

You can suppress the writing of an output data set by specifying particular installation-defined class defined to delete the data set before it is printed. This technique is used by the installation to suppress the output of started tasks such as START and MOUNT commands. You can also suppress the writing of an output data set by specifying COPIES=0 on the FORMAT PR or PU (print or punch) statements.

If the device on which the data set will be written is specified in the UNIT parameter, you can assign the data set a dummy status by coding DUMMY or by assigning the data set name NULLFILE. All parameters other than DUMMY or DSNAME=NULLFILE and DCB are ignored; no units are assigned to the data set. When the program requests that the data set be written, the request is recognized but no data is transmitted. The facility is available by use of the basic sequential access method (BSAM) or queued sequential access method (QSAM) in a request to write a dummy data set. If any other access method is used, the job is terminated.

Limiting Output Records

The number of logical records in the output data set can be limited by specifying a maximum number of records through the use of the OUTLIM parameter on the DD statement. For example, a program is printing and goes into an endless loop. You anticipate the problem and have a maximum number of records printed before the system discontinues the output processing.

To limit the printed or punched output of a job, specify the estimated number of lines of output or the estimated number of cards associated with your job's output by coding the LINES and CARDS parameters on the MAIN statement. This information is used by JES3 to monitor output and take whatever action is specified if you exceed the estimates. These actions request that the operator receive a warning (the WARNING subparameter), that the job is canceled (the CANCEL subparameter), or that the job is canceled with a storage dump (the DUMP subparameter). JES3 initialization parameter values are used if you omit the estimates.

The LINES parameter will not limit the size of an internal reader data set because it is not considered to be part of the printed output of a job. To restrict this type of data set, the OUTLIM parameter must be specified on the DD statement.

Requesting Multiple Copies of an Output Data Set

You can control the number of hard copies produced by the SYSOUT data sets. You can request as many as 254 copies of an output data set by coding the COPIES parameter on the SYSOUT DD statement defining the data set or up to 255 copies by coding the COPIES parameter on the JES3 FORMAT PR control statement.

With the 3800 Printing Subsystem, you can also specify on the SYSOUT DD statement or on the FORMAT PR statement how the copies of the output data set are to be grouped. Each group value of the COPIES parameter specifies the number of copies of each individual page that is to be printed before copies of the next page are printed. The total number of copies printed equals the sum total of the group values.

Requesting Copy Modification

Selected copies of output can be modified when using the 3800 Printing Subsystem by specifying a copy modification module name on the MODIFY parameter of the SYSOUT or output DD statement or on the JES3 FORMAT PR statement. This allows the printing of predefined data on all pages of a specific copy or copies of a data set. For example, you may want to vary column headings or explanatory remarks on different copies of the same printed page of data. Copies might also be personalized with the recipient's name, address, and other desired information. Blanks or printable characters, such as asterisks, might also be used to suppress the printing of variable data on particular copies of a page. (This is a function done in other printers by using short or spot carbon in the forms set.)

The predefined data is created as a copy modification module and stored on SYS1.IMAGELIB using the IEBIMAGE utility program. For information on using IEBIMAGE, see the **IBM 3800 Printing Subsystem Programmer's Guide.**

Requesting Printer Form and Character Control

When requesting that an output data set be written, you can give JES3 special instructions on how to write the data set. You can request:

- A special output form.
- A special character set or arrangement, when output is being printed by a 3211, 1403, or 3203-5 printer with the universal character set feature or by a 3800 Printing Subsystem.
- A specific FCB (forms control buffer) module, which controls how many lines per inch are printed and the length of the form, when the data is written to a remote job processing (RJP) printer supported by systems network architecture (SNA) or to a 3211 or 3203-5 printer or a 3800 Printing Subsystem.
- A specific carriage control tape, when the data set is written to a 1403 printer.
- A test for printer overflow and spacing.
- Interpretation of punch output on the 3525.

Requesting a Special Output Form

Special forms are requested for output data set printing by including the form name in the SYSOUT parameter on the DD statement defining the data set or by coding the FORMS parameter on the FORMAT PR control statement. For example, assign a data set to an output class to be routed to a printer and specify the data set be printed on a special form. (For example, code SYSOUT=(A,,FMS2).) JES3 and the external writer insure that the proper form is mounted.

Requesting a Special Character Set Using the UCS Feature

The universal character set (UCS) feature is requested by coding the UCS parameter on a DD statement defining an output or SYSOUT data set or by coding the TRAIN parameter on the FORMAT PR control statement for SYSOUT data sets. You can request the UCS feature for different sets of characters to be printed for various applications.

To request a special character set for a 3211, 3203-5, or 1403 printer, specify the code identifying the character set in the UCS parameter or the FORMAT statement. The codes for the IBM standard special character sets are in Figure 12.

1403	3203-5	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 alphameric character set, arrangement A
PCHN	PCHN		Preferred alphameric character set, arrangement H
PN	PN	P11	PL/I alphameric character set
QN	QN		PL/I preferred alphameric character set for scientific applications
QNC	QNC		PL/1 preferred alphameric 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	XN		High-speed alphameric character set for 1403, Model 2
YN	YN		High-speed preferred alphameric character set for 1403, Model 3 or N1

Figure 12. Special Character Sets for the 1403, 3203-5, and 3211 Printers (JES3)

Note: Where three values exist (for the 1403, 3211, or 3203-5 printers), either can be coded and JES3 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. In addition, the installation can design character sets to meet special needs and assign a unique code to them. See the system programming staff for a list of character sets for the installation.

Requesting Character Arrangements with a 3800 Printing Subsystem

Character arrangement tables to be used when printing with the 3800 are specified with the CHARS parameter on the SYSOUT or output DD statement or on the JES3 FORMAT PR statement. The table names supplied for the 3800 are given in the **IBM 3800 Printing Subsystem Programmer's Guide**. See your system programmer for the selection of table names available at your installation.

When more than one character arrangement table is specified, you can code OPTCD=J as a DCB subparameter to indicate that your data line contains a table reference character for dynamically selecting the table you want. (See the description of the OPTCD subparameter for BSAM and QSAM in the topic, "The DCB Parameter".) Using the IEBIMAGE utility program, you can modify or construct character arrangement tables and graphic character modification modules to allow substitution of existing or user-designed characters. Details for using both IEBIMAGE and the OPTCD subparameter are provided in the **IBM 3800 Printing Subsystem Programmer's Guide.**

The UCS (universal character set) parameter can be specified on the same output DD statement with the CHARS parameter to permit output to go to either the 3800 or to other printers. The GF10, GF12, GFC, or GF15 character arrangement table coded on the CHARS parameter provides the same effect as the FOLD subparameter of UCS. If a printer other than the 3800 is allocated, the CHARS parameter is ignored.

Requesting Forms Control

For a 1403 Printer: Forms control is requested by specifying a specific carriage control tape in the CARRIAGE parameter on the FORMAT PR control statement. Carriage specifications are used for JES3 output processing only; they are ignored by the external writer.

For 3211 or 3203-5 Printers and Printers Supported by Systems Network Architecture (SNA) Remote Job Processing (RJP): Specific forms control images (for example, the number of lines per page or number of characters per line) are requested by coding an image identifier in the FCB parameter on a DD statement or on the FORMAT PR control statement. A carriage control tape for JES3 output processing only can also be specified in the CARRIAGE parameter on the FORMAT PR control statement. The FCB image is stored on SYS1.IMAGELIB. IBM provides two standard FCB images: STD1 and STD2. STD1 specifies that 6 lines per inch are to be printed on an 8.5-inch form. STD2 specifies that 6 lines per inch are to be printed on an 11-inch form. (Do not specify STD1 or STD2 for JES3 processing unless instructed by your installation.) Additional FCB images can be specified by the installation. For information on IBM- and user-supplied FCB images, see **OS/VS2 System Programming Library: Data Management.**

For a 3800 Printing Subsystem: Forms control is requested by specifying an FCB module name in the FCB parameter on a DD statement or on the FORMAT PR control statement. (Although the FCB image for the 3211 and the FCB module for the 3800 serve the same purpose, they are constructed differently and are not interchangeable between the two printers.) The FCB module is stored on SYS1.IMAGELIB. IBM provides a standard FCB module, STD3, which specifies output of 80 lines per page at 8 lines per inch on 11-inch long paper. (For a 3800 using ISO paper sizes, STD3 can be redefined by the installation.) Additional FCB modules can be specified by the installation. For information on IBM- and user-supplied FCB modules, see the **IBM 3800 Printing Subsystem Programmer's Guide.**

Requesting Forms Overflow and Printer Spacing

Use the JES3 overflow subparameter (OVFL) to prevent printing across page folds. Specifying OVFL=ON on the //*FORMAT PR statement or the JES3 SYSOUT initialization statement causes the printer to eject a page when it senses the end-of-forms indicator (channel 12) on the printer's carriage control or in the printer's FCB.

Page ejection can also be controlled by specifying the CONTROL=PROGRAM subparameter on the //*FORMAT PR statement. This causes the format specified in the DCB=(...,RECFM=format,...) subparameter on the DD statement to be used for printer carriage control. This method must not be used for printer control (tape or RCB) contains channel 12 indicators and OVFL=ON (the default).

JES3 defaults to OVFL=ON and CONTROL=PROGRAM on the //*FORMAT PR statement. Therefore, you must specify OVFL=OFF on the //*FORMAT PR statements for data sets that are program controlled. You can turn overflow off by specifying OVFL=OFF on each data set's //*FORMAT PR statement.

If the number of data sets requiring OVFL=ON is small, the installation can turn off the overflow by specifying OVFL=OFF on the SYSOUT initialization statement. For those data sets requiring overflow, code a //*FORMAT PR statement with OVFL=ON to override the SYSOUT initialization statement.

Requesting Punch Output Interpretation on a 3525

Punched output may or may not be interpreted depending on the installation-defined standard for the SYSOUT class. You can specify punched output to be interpreted by coding the INT=YES parameter on the JES3 FORMAT PU statement. If you omit the device name that specifies a 3525I, JES3 attempts to find one for the output. If you specify a non-interpreting punch device, output is punched on it but not interpreted.

Cards punched on a 3525 card punch from output spooled by JES3 will be interpreted if you code FUNC=I as a DCB subparameter on the SYSOUT card and if the spooled output is processed by a JES3 writer rather than the external writer. The FUNC=I subparameter will be ignored if the spooled output is processed by the JES3 writer onto a card punch other than the 3525. You should check with the installation to determine if a special output class has been set aside for 3525 output. Card interpretation by the external writer is an operator specified function. Output to be interpreted should be placed in a class designated by the installation as a punch with interpretation class.

Requesting Forms Overlay

The forms overlay feature of the 3800 Printing Subsystem allows printing of the image from a forms overlay negative together with the data being printed. This reduces the need for pre-printed forms, and for changing of forms. The FLASH parameter on the DD statement or on the JES3 FORMAT PR statement identifies the overlay to be used and the number of copies on which that overlay is to be printed. For information on designing and making or obtaining forms overlay negatives, see the **Forms Design Reference Guide for the IBM 3800 Printing Subsystem.**

Bursting of Output

The optional Burster-Trimmer-Stacker of the 3800 Printing Subsystem separates continuous form paper into individual sheets. The BURST parameter on the DD statement or the STACKER parameter on the JES3 FORMAT PR statement is used to specify to the operator whether the output is to go to the Burster-Trimmer-Stacker or to the continuous forms stacker. For further information and examples, see the topic, "The BURST Parameter".

Controlling Output Destination

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

When submitting a job from a local CPU or a work station, the output is returned to the place where it is submitted unless you code ORG or you specifically route the output; simply assign output data sets to an output class (with the SYSOUT parameter) and messages from the job to an output class (with the MSGCLASS parameter). At remote stations, JES3 offers most of the same options for writing data sets that are requested when submitting the job at the central computing center. You can request:

- That a data set be held until the operator requests that it be printed.
- That a special output form be used by specifying a form name in the SYSOUT parameter.
- That multiple copies of the data set be used.

Whether at a remote station or at the central computing center, you can also request that a data set be routed to another destination. To route an output data set to another destination, code the identification of that destination in the DEST parameter on the DD statement defining the data set, code the MAIN ORG statement, or code the FORMAT PR or PU DEST parameters. Work stations are identified by a destination identification that has been established by the

system programmer. The DEST parameter on the DD statement and the DEST parameter on the FORMAT PR and PU statements route individual data sets to a remote destination (work station), a local destination (central computing center), or a specific local device.

Remote Job Processing

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

Routing output to other destinations is also possible using the DEST parameter on the FORMAT statement and the DD statement. Refer to the previous section for more information.

Controlling Job and Output Destination Using JES3 Networking

JES3 networking allows users at one JES3 node to send jobs to another node for execution, or to send output to other nodes for processing. Additionally, users at a JES3 node can send data to nodes to which the sending node does not have a direct connection by way of an intermediate node. To control routing within the network, you specifically designate any jobs or SYSOUT data to be sent to other nodes with a JES3 //*ROUTE XEQ control statement. A //*ROUTE XEQ statement encountered after a local JOB statement, causes JES3 to route the next job to the node specified on the //*ROUTE XEQ control statement. A //*ROUTE XEQ control statement has the format:

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

The nodename refers to the JES3 global processor and all of its local processors. The nodename cannot refer to an individual processor within that JES3 node. If the nodename specifies the node on which the job was entered, the first JOB statement and the //*ROUTE XEQ statement will be flushed and the job that follows will execute on the node of entry. You can also specify a VM/370 userid as a secondary destination on this control statement.

You can route SYSOUT data to other nodes in the network using the ORG parameter on the //*MAIN control statement and/or the DEST parameter on the //*FORMAT control statement. In addition, you can have SYSOUT data processed by an external writer at the destination node by specifying the EXTWTR parameter on the //*FORMAT control statement.

Example of Obtaining Output (JES3)

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

```
//OUTJOB     JOB      BAKER,PERFORM=100,MSGCLASS=J
//*FORMAT    PR,DDNAME=,COPIES=2,FORMS=GRN1
//*FORMAT    PR,DDNAME=DD3,DEST=PRINTER8,CARRIAGE=STD3,
//*FORMS=2PRT,TRAIN=TN
//STEP1      EXEC     PGM=TESTSYSO
//DD1        DD       DSN=DATA,UNIT=2314,VOL=SER=SCHLIB,
//                    DISP=(OLD,KEEP),SPACE=(TRK,(5,2))
//DD2        DD       DSN=&TEMP,UNIT=2314,DISP=(NEW,DELETE),
//                    SPACE=(TRK,(10,5))
//DD3        DD       SYSOUT=(A)
//DD4        DD       SYSOUT=(A,,GRPH)
//DD5        DD       SYSOUT=L
```

1. The job will run in performance group 100; the meaning of 100 is defined by the installation. All system messages are to be written to output class J.
2. The first FORMAT statement indicates that:
   a. All print data sets (according to class) with no FORMAT statements will be printed according to the parameters on this statement unless the output class defines specific processing characteristics (DDNAME is coded without a name).
   b. Two copies are printed.
   c. Forms name GRN1 and two copies are to be used by all data sets unless a specific form or number of copies is defined on a DD statement or by class by the installation.
3. The second FORMAT statement indicates that:
   a. The destination for the output is a printer that has an installation-defined name of PRINTER8.
   b. If PRINTER8 has the forms control buffer feature, STD3 defines the special forms control buffer image or carriage tape to be used for processing the job.
   c. Forms name 2PRT is the name of the forms for DD3.
   d. TN means text printing on a 1403, 3211, or 3203-5 printer.

Data sets can be defined to satisfy a special purpose. Such data sets are usually defined with a special ddname, a specific data set name, or a specific parameter.

This section includes eight topics:

- Creating and Using Private and Temporary Libraries
- Defining a Dummy Data Set
- Using Virtual Input/Output (VIO) for Temporary Data Sets
- Entering Data Through the Input Stream
- VSAM Data Sets
- Creating and Retrieving Indexed Sequential Data Sets
- Creating and Retrieving Generation Data Sets
- Creating and Using a Subsystem Data Set

Creating and Using Private and Temporary Libraries

A library is simply a partitioned data set — a data set in direct access storage that is divided into partitions, called members, each of which can contain a program or part of a program. Each partitioned data set contains a directory (or index) that the control program can use to locate a program in the library. All programs that can be executed must exist in a library; that is, they must be members of a partitioned data set.

A private library is a partitioned data set that contains user-written programs. You inform the system that a program exists in a private library by coding a DD statement defining that library. You can define a private library to be used throughout the job by coding a DD statement with the ddname JOBLIB, or define a library to be used in a specific step by coding a DD statement with the ddname STEPLIB. The library defined by a JOBLIB or STEPLIB DD statement is searched prior to the system libraries (such as SYS1.LINKLIB for the program to be executed (that is, the program named in the PGM= field of the EXEC statement).

A temporary library is a partitioned data set created in the job to store a program, as a member of the partitioned data set, until it is executed in a following step. For example, if in the job you want to assemble, linkage edit, and then execute a program, make the output of the linkage editor a member of a library. Any library that is created and deleted in the same job is a temporary library.

Code the PGM parameter as the first parameter on the EXEC statement to execute a program contained in a library. If the program exists in a private library, code PGM=program name and either a JOBLIB or STEPLIB DD statement. If the program exists in a temporary library, code either PGM=*.stepname.ddname or PGM=*.stepname.procstepname.ddname. Ddname is a temporary library created in and pointed to by stepname and procstepname. They identify the job step or job step and procedure step defining the library. If you define a private library, the system looks in that library for the program you want executed.

This chapter describes how to code JCL statements to create or retrieve private and temporary libraries. Complete information on creating a partitioned data set, adding members to and deleting members from a partitioned data set, is included in **OS/VS2 MVS Data Management Services Guide.**

Creating a Private Library

Use the JOBLIB DD statement to create a private library. The JOBLIB DD statement must appear immediately after the JOB statement and any JES statements. Do not use the ddname

JOBLIB unless you are defining a private library. The library defined with a JOBLIB DD statement is automatically available to every step in the job. (The STEPLIB DD statement is included among the DD statements in a step and is available only to that step unless you pass the library or redefine it in subsequent steps; since the library on a JOBLIB DD statement is available to every step, it is easier to create a library with the JOBLIB DD statement.)

When creating the library on the JOBLIB DD statement, you are creating a partitioned data set. Steps in the job must add members to the library before those members (programs) can be used by subsequent steps.

On the JOBLIB DD statement, assign the library a name in the DSNAME parameter, give unit and volume information in the UNIT and VOLUME parameters (a partitioned data set must be contained on one direct access volume; if, however, you make a nonspecific volume request, you need not code the VOLUME parameter), request space for the entire library in the SPACE parameter, and assign a data set status and disposition in the DISP parameter. Code NEW as the data set status and either CATLG or PASS as the data set disposition. When specifying CATLG, the library is cataloged, available throughout the job, and kept at the end of the job. When specifying PASS, the library is available throughout the job, but is deleted at job termination. (If you do not code a disposition, or code a disposition other than CATLG or PASS, the system assumes DELETE. This means that the library will be deleted at the end of the first step and will not be available to any later job steps.) You must also code the DCB parameter if complete data control block information is not included in the data set label.

Adding Members to a Private Library

Add members to the library in job steps within the job by coding a DD statement that defines the library and names the member to be added 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). Do not code the SPACE parameter; request space for the entire library on the JOBLIB DD statement. In the DISP parameter, specify MOD as the data set status; the partitioned data set already exists since you created it in the JOBLIB statement, and you are lengthening it with a new member. If you cataloged the library in the JOBLIB DD statement, that is, coded DISP=(NEW,CATLG), do not respecify CATLG when adding a member: you need not code a second disposition at all. For a cataloged library, you do not have to specify unit and volume information, except in one instance: if you are adding a member to the library in the first step of the job, supply unit and volume information; the library is not cataloged until the first step completes the execution. Refer to the JOBLIB DD statement for unit and volume information by coding VOL=REF=*.JOBLIB.

In the following example, the JOBLIB DD statement creates a library named GROUPLIB; STEP1 adds the program RATE to the library; STEP2 calls the program RATE:

```
//EG          JOB     MSGLEVEL=1
//JOBLIB      DD      DSNAME=GROUPLIB,DISP=(NEW,CATLG),
//                    UNIT=2314,VOL=SER=727104,
//                    SPACE=(CYL,(50,3,4))
//STEP1       EXEC    PGM=FIND
//ADDPGMD     DD      DSNAME=GROUPLIB(RATE),DISP=MOD,
//                    VOL=REF=*.JOBLIB
//STEP2       EXEC    PGM=RATE
```

In STEP1, the system looks for the program named FIND in SYS1.LINKLIB — the private library created on the JOBLIB DD statement does not actually exist until a member is added to it. In STEP2, the system looks for the program named RATE first in the private library.

Retrieving an Existing Private Library

If you are retrieving several programs from one library (several steps in the job will be using the library), use the JOBLIB DD statement to define the library: the library will be available in every step of the job for which you do not code a STEPLIB DD statement. The JOBLIB DD statement must appear immediately after the JOB statement. To make a library available in a single step, define the library on a STEPLIB DD statement. The STEPLIB DD statement is included with the DD statements for a step (in no specific order) and is available only to that step, unless you pass the library and retrieve it in a subsequent step. Use the ddnames JOBLIB and STEPLIB only when defining private libraries.

The system will search for a program in the private library you define. If both JOBLIB and STEPLIB DD statements appear in a job, the STEPLIB definition has precedence, that is, the private library defined by the JOBLIB DD statement is not searched for any step that contains the STEPLIB definition. If you want JOBLIB definition ignored but the step does not require use of another private library, define a system library on the STEPLIB DD statement:

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

Retrieve a private library as you would any partitioned data set: if the library is cataloged, or in the case of a STEPLIB definition, passed from a previous step, you need not specify unit and volume information; otherwise, you must code the UNIT and VOLUME parameters.

For both cataloged and uncataloged libraries, code: the DSNAME parameter, specifying the name of the library; the DCB parameter, if complete data control block information is not included in the data set label; and the DISP parameter, specifying data set status and disposition. Normally, you will want to specify SHR as the data set status: SHR indicates that the data set is old, but also allows other jobs to simultaneously use the library. All references to the library in the job must specify SHR if the data set is to be shared; do not code SHR, however, if you will be adding members to the library in the job. (A more thorough discussion of sharing a data set is included in the chapter "Insuring Data Set Integrity.") Code PASS as the data set disposition for a library defined on the JOBLIB DD statement: PASS makes the library available throughout the job. (If you do not code a disposition, the system assumes PASS.) For a library defined on a STEPLIB DD statement, code any valid disposition, depending on how you want the data set treated after its use in the job step: for example, if the library is not cataloged, and you want it to be cataloged, code CATLG; if you want the library deleted, code DELETE.

The following job includes both JOBLIB DD and STEPLIB DD statements:

```
//CAMILLE   JOB     MSGLEVEL=1
//JOBLIB    DD      DSNAME=LIB5.GRP4,DISP=SHR
//STEP1     EXEC    PGM=FIND
//STEP2     EXEC    PGM=GATHER
//STEPLIB   DD      DSNAME=ACCOUNTS,DISP=(SHR,KEEP),
//                  UNIT=2314,VOL=SER=727104
```

In STEP1, the system searches the library named LIB5.GRP4, defined on the JOBLIB DD statement, for the program named FIND. In STEP2, the system searches the library named ACCOUNTS, defined on the STEPLIB DD statement, for the program named GATHER.

Add a program to an existing library by coding a DD statement in a job step that defines the library and names the program to be added — see "Adding Members to a Private Library" for details on coding this DD statement. The new member must be added to the library before it can be executed (the step that adds the program to the library must precede the step that calls the program). Do not code SHR as the data set's status when modifying the library.

Concatenating Private Libraries

If the job uses programs contained in several libraries, you can concatenate these libraries on one JOBLIB DD statement or one STEPLIB DD statement; all the libraries concatenated must be existing libraries. Omit the ddname from all the DD statements defining the libraries, except the first:

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

This entire group must appear immediately after the JOB statement. When concatenating libraries using STEPLIB as the ddname, the entire group appears as part of the DD statements for the step.

The system will search the libraries for the program in the order in which the DD statements defining the libraries are coded.

Using Private Catalogs

Use Access Method Services to define private user catalogs, as explained in **OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide**. A JOBCAT or STEPCAT is a private catalog which is searched prior to the system catalog whenever a DD statement does not specify unit and volume serial information for a data set. JOBCAT applies to each step of a job in which a STEPCAT has not been specified. To locate a data set, VSAM searches catalogs in the following order:

1. User catalogs specified in the current job step (STEPCAT), or user catalogs specified in the current job (JOBCAT), if no user catalogs are specified for the job step.
2. A CVOL or user catalog indicated by the first qualifier of the data set name, if any.
3. The master catalog.

Temporary Libraries

Temporary libraries are libraries that are created and deleted within the job. It is not necessary to define a temporary library on a JOBLIB DD or STEPLIB DD statement: simply code a DD statement creating a partitioned data set and adding the program to it in the step that produces the program. You can then retrieve this program in a subsequent step. (You can also use the VIO facilities to define temporary data sets. For more information, refer to "Defining a VIO Temporary Data Set" later in this section.)

For example, STEP2 illustrated below calls the program IEWL, which linkage edits object modules to form a load module that can be executed. Place the results of the linkage edit step in a library so that a subsequent step can use those results. Since the results are not a program other jobs will call, it is logical to place the program in a temporary library:

```
//STEP2     EXEC    PGM=IEWL
//SYSLMOD   DD      DSNAME=&&PARTDS(PROG),UNIT=2314,
//                  DISP=(NEW,PASS),SPACE=(1024,(50,20,1))
//STEP3     EXEC    PGM=*.STEP2.SYSLMOD
```

Call the program in STEP3 by naming the step in which the library was created and the name of the DD statement that defines the program as a member of a library. If STEP2 had called a procedure, and the DD statement named SYSLMOD was included in PROCSTEP3 of the procedure, you would code PGM=*.STEP2.PROCSTEP3.SYSLMOD.

Defining a Dummy Data Set

To save processing time, you might not want a data set to be processed every time the job is executed. For example, while testing a program, you might want to suppress the writing of an output data set until you are sure it will contain meaningful output; you might want to skip the reading of a data set to be used only once a week. When defining a dummy data set, input/output operations are bypassed, disposition processing is not performed, and devices and storage are not allocated to the data set.

Define a dummy data set by:

- Coding the DUMMY parameter on the DD statement.
- Assigning the data set name NULLFILE in the DSNAME parameter on the DD statement.

Coding the DUMMY Parameter

Code DUMMY as the first parameter on the DD statement. DUMMY is a positional parameter: it must precede all keyword parameters on the DD statement.

When the DUMMY parameter is coded, all other parameters on the DD statement, with the exception of the DCB parameter, are ignored. (The parameters are checked for syntax, however; if a parameter is coded incorrectly, a JCL error message is issued.) Therefore, although you can code UNIT, VOLUME, and DISP, no devices or external storage is allocated to the data set and no disposition processing is performed. The DCB parameter must be coded if you would code it for normal I/O operations. For example, when an OPEN routine requires a BLKSIZE specification to obtain buffers and BLKSIZE is not specified in the DCB macro instruction, you should supply this information in the DCB parameter on the DD statement.

When a DD statement that overrides a procedure DD statement contains the DUMMY parameter, all of the parameters coded on the procedure DD statement are nullified, except for the DCB parameter.

If you request unit or volume affinity with a dummy data set, the data set requesting affinity is assigned a dummy status. (Unit and volume affinity is described in the chapter "Requesting Units and Volumes.")

When you want the data set to be processed, replace the DD statement containing the DUMMY parameter with a DD statement containing the parameters required to define the data set. When a procedure DD statement contains the DUMMY parameter, nullify it by coding the DSNAME parameter on the overriding DD statement and assigning a data set name other than NULLFILE.

Coding DSNAME=NULLFILE

Assigning the name NULLFILE in the DSNAME parameter has the same effect as coding DUMMY. The data set is assigned a dummy status; no device or storage is allocated and no disposition processing is performed. All parameters except for DSNAME and DCB are ignored. (The parameters are checked for syntax, however; if a parameter is coded incorrectly, a JCL error message is issued.) Code the DCB parameter when defining a dummy data set if you would code it for normal I/O operations.

When you want the data set to be processed, replace the name NULLFILE with another data set name. (Assigning names to data sets is described under "Specifying the DSNAME Parameter.")

Requests to Read or Write a Dummy Data Set

When the program asks to read a dummy data set, an end-of-data-set exit is taken immediately. When the program requests that the data set be written, the request is recognized but no data is transmitted. VSAM supports dummy data sets for both read and write processing. Otherwise, use the basic sequential access method (BSAM) or queued sequential access method (QSAM) when requesting to write a dummy data set; if any other access method is used, the job is terminated.

If you define a data set as a dummy data set, the DISP parameter, if coded, is ignored and disposition processing is not performed.

Using Virtual Input/Output (VIO) for Temporary Data Sets

Temporary data sets can be handled by a new facility called virtual I/O (VIO). (VIO processing does not apply to nontemporary data sets.) Data sets for which VIO is specified reside within the paging space; however, to a problem program and the access method, the data sets appear to reside on some other real direct access storage device.

During system generation, new and/or existing unit names can be defined as eligible for VIO. These unit names can be coded on a DD statement defining a data set to specify VIO processing for any system-named temporary data set.

Defining a VIO Temporary Data Set

The DD statement for a VIO data set is similar to the DD statement for a conventional temporary data set, with the following exceptions:

- The UNIT keyword in the VIO DD statement must specify a name that has been defined as eligible for VIO.
- If the SPACE parameter is not coded for virtual I/O data sets, the default value is 10 primary and 50 secondary blocks with an average block length of 1000. Up to a one volume limit, you will always obtain the full amount of space requested (that is, the primary quantity plus fifteen secondary requests). If the primary quantity for space is larger than the simulated volume, the job will fail. If the primary request is met, but the secondary request is greater than one volume, you will get up to one volume. When allocating by average block length for a VIO data set, the secondary request is determined by the average block length specified in the SPACE parameter.
- VIO does not support ISAM or VSAM, so you can not specify ISAM or VSAM indicators in the DSORG parameter of a DD statement for a VIO data set. The "area" of an ISAM data set cannot be specified in the DSNAME parameter.
- The DISP parameter must be specified as NEW or PASS when creating a data set. Do not specify KEEP or CATLG any time for the DISP parameter.
- The DSNAME parameter need not be coded, but if it is, it must only be specified in & & name form.
- Volume serial numbers cannot be specified for VIO. A VIO data set will be allocated to non-VIO if any of the above exceptions are violated, except the SPACE parameter request.
- The unit count subparameter of the UNIT parameter is ignored.

Note: Empty input data sets and SUBALLOC requests are not eligible for VIO.

Backward References to VIO Data Sets

If the referring DD statement (VOL=REF=) defines a temporary data set and refers to a DD statement that defines a VIO data set, the data set is assigned to external page storage as a VIO data set.

If the referring DD statement requests unit affinity but does not define a temporary data set, the referring statement assumes the unit specification of the DD statement to which reference is made, but not the VIO status.

The following examples assume that the user-assigned group name SYSDA and the device type name 3330 have been defined at system generation with the UNITNAME macro instruction as group names eligible for VIO processing.

The data sets defined by the following DD statements are assigned to external page storage for VIO processing:

```
//DD1       DD      UNIT=SYSDA
```

```
//DD2       DD      UNIT=3330
```

```
//DD3       DD      DSN=&&A,DISP=(NEW),SPACE=(CYL,(30,10)),UNIT=SYSDA
```

```
//DD1       DD      UNIT=SYSDA
//DD2       DD      VOL=REF=*.DD1
```

```
//DDA       DD      UNIT=SYSDA
//DDB       DD      VOL=REF=*.DDA,UNIT=3330
```

In each of the following examples, the data set defined on the first DD statement is assigned to external page storage for VIO processing. The second DD statement does not request VIO because it defines a nontemporary data set.

```
//DD1       DD      UNIT=SYSDA
//DD2       DD      DSN=NONTEMP,DISP=(,KEEP),
//                  VOL=REF=*.DD1,SPACE=(CYL,10)
```

```
//DD1       DD      UNIT=SYSDA
//DD2       DD      DSN=TEMP,DISP=(,KEEP),VOL=SER=665431,
//                  SPACE=(CYL,10),UNIT=AFF=DD1
```

Using Virtual Input/Output (VIO) to Pass Temporary Data Sets Among Job Steps

VIO data sets are passed the same as conventional data sets. For example, the following JCL statements show the DD statements required by VIO for a job with compilation, linkage editor, and go steps. The VIO data sets in the various job steps are defined as system-named temporary data sets. The unit name PAGEDEV has been defined as eligible for VIO with the UNITNAME macro instruction during system generation.

```
(1) //ASM         EXEC    PGM=IFOX00
    ---
    ---
    //ASM.SYSGO DD        DSN=&&OBJ,UNIT=PAGEDEV,DISP=(NEW,PASS)
(2) //LKED        EXEC    PGM=IEWL
    //SYSLIN      DD      DSN=&&OBJ,DISP=(OLD,DELETE)
    //            DD      DDNAME=SYSIN
    //SYSLMOD     DD      DSN=&&LOAD(A),DISP=(NEW,PASS),UNIT=PAGEDEV,
    //                    DCB=DSORG=PO,SPACE=(TRK,(5,5,1))
    ---
    ---
(3) //GO          EXEC    PGM=*.LKED.SYSLMOD
```

Note: You must code the SPACE parameter on the //SYSLMOD DD card to ensure that directory space is allocated.

Entering Data Through the Input Stream

You can enter data through the input stream by coding either the * or DATA parameters on the DD statements. The DD * statement precedes data in an input stream; the DD DATA statement precedes data in an input stream when the data contains JCL statements. The DLM parameter allows the use of a delimiter other than /* to terminate data defined in the input stream. Code this parameter on either the DD * or DD DATA parameters.

You can include several distinct groups of data in the input stream. Two types of data are for job steps specifying a program name or for job steps that call a cataloged or in-stream procedure. However, cataloged and in-stream procedures cannot contain DD statements defining data in the input stream.

VSAM Data Sets

Virtual Storage Access Method (VSAM) is an access method of OS/VS for use with direct-access storage. It is different from all other access methods and you need to take certain precautions when coding VSAM data sets. You can use JCL parameters to identify cataloged VSAM data sets and to specify options for them. To process a VSAM data set, specify a DD statement in the form:

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

The DSNAME parameter specifies the name of the cluster to which the data set you are processing belongs. The DISP parameter must specify either OLD or SHR because the data set is cataloged. You cannot use JCL to create VSAM data sets; you must use Access Method Services commands. VSAM data sets cannot be passed within a job.

Some DD parameters and subparameters have different meanings for VSAM data sets. For example, VSAM data sets are described by the access-method control block (ACB), not the DCB. Therefore, the DCB parameter is not applicable to VSAM. Parameters that can be used without modification are explained in Figure 13; parameters that either should not be used or should be used only with caution are explained in Figure 14. The STEPCAT and JOBCAT facilities identify user catalogs. These parameters are similarly used for all data sets and are discussed in this section under "Creating and Using Private Libraries."

VSAM has one JCL parameter of its own: AMP. The AMP parameter takes effect when the data set defined by the DD statement is opened. It has subparameters for:

- Overriding operands specified with the ACB, EXLST, or the GENCB macro instructions.
- Supplying operands missing from the ACB or GENCB macro instruction.
- Indicating checkpoint/restart options.
- Indicating options when using ISAM macro instructions to process a key-sequenced data set.
- Indicating that the data set is a VSAM data set when you specify unit and volume information or DUMMY in a DD statement.
- Indicating that you want VSAM to supply storage dumps of the access-method control block(s) that identify this DD statement.

Parameter	Subparameter	Comment
DDNAME	*ddname*	Works as in OS/VS.
DISP	**SHR**	Indicates that you are willing to share the data set with other jobs. This subparameter alone, however, does not guarantee that sharing will take place. See *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* for a full description of data-set sharing.
	OLD	Works as in OS/VS.
DSNAME	*dsname*	Works as in OS/VS.
DUMMY		Works as in OS/VS, except that an attempt to read results in an end-of-data condition, and an attempt to write results in a return code that indicates the write was successful. If specified, AMP=AMORG must also be specified.
UNIT	*address*	Must be the address of a valid device for VSAM (2305, 2314, 2319, 3330V, 3330, 3340, 3344, 3350, 3375, or 3380). If not, OPEN will fail
	type	Must be a type supported by VSAM (2305, 2314, 3330, 3330V, 3340, 3350, 3375, or 3380). If not, OPEN will fail.
	group	Must be a group supported by VSAM. If not, OPEN will fail.
	p	There must be enough units to mount all of the volumes specified. If sufficient units are available, UNIT=p can improve performance by avoiding the mounting and demounting of volumes.
	unit count	If the number of devices requested is greater than the number of volumes on which the data set resides, the extra devices are allocated anyway. If a key-sequenced data set and its index reside on unlike devices, the extra devices are allocated evenly between the unlike device types. If the number of devices requested is less than the number of volumes on which the data set resides but greater than the minimum number required to gain access to the data set, the devices over the minimum are allocated evenly between unlike device types. If devices beyond the count specified are in use by another task but are shareable and have mounted on them volumes containing parts of the data set to be processed, they will also be allocated to this data set.
	DEFER	Works as in OS/VS.
VOLUME	**PRIVATE**	Works as in OS/VS.
	SER	The volume serial number(s) used in the Access Method Services DEFINE command for the data set must match the volume serial numbers in the VOLUME=SER specification when the data set is defined. After a VSAM data set is defined, the volume serial number(s) need not be specified on a DD statement to retrieve or process the data set. If, however, VOLUME=SER and UNIT=type are specified, only those volumes specifically named are initially mounted. Other volumes may be mounted when they're needed if at least one of the units allocated to the data set is not shareable or the unit count is equal to the total number of volumes allocated to the data set. A unit is unshareable when unit count is less than the number of volume serial numbers specified or when DEFER is specified. If VOLUME=SER is specified and the data set is cataloged in a user catalog, the user catalog should be defined as a JOBCAT or a STEPCAT for the current step.

Figure 13. DD parameters used with VSAM

Parameter	Subparameter	Comment
DATA		Because there is no way to get VSAM data into the input stream. this parameter is not applicable to VSAM.
DCB	All	The access-method control block, not the DCB, describes VSAM data sets; therefore, the DCB parameter is not applicable to VSAM. An access-method control block is generated by an ACB or GENCB macro, and can be modified by a MODCB macro.
DISP	**CATLG**	VSAM data sets are cataloged and uncataloged as a result of an Acess Method Services command; if CATLG is coded, a message is issued, but the data set is not cataloged.
	DELETE	VSAM data sets are deleted as a result of an Acess Method Services command; if DELETE is coded, a message is issued, but the data set is not deleted.
	MOD	For VSAM data sets, MOD is treated as if OLD were specified, except for processing with an ISAM program, in which case MOD indicates resume load.
	KEEP	Because KEEP is implied for VSAM data sets, it need not be coded.
	NEW	VSAM data spaces are initially allocated as a result of the Access Method Services DEFINE command. If NEW is specified, OS/VS also allocates space, and it is never used by VSAM. Moreover, an Access Method Services request for space may fail if the DISP=NEW acquisition of space causes too little space to remain available.
	UNCATLG	VSAM data sets are cataloged and uncataloged as a result of Access Method Services commands; if UNCATLG is coded, a message is issued, but the data set is not uncataloged.
	PASS	The PASS parameter is not applicable to VSAM. However, because there is no error checking, coding PASS for a key-sequenced data set whose index resides on a like device does not result in an error. If a VSAM data set and its index reside on unlike devices, the results are unpredictable. In either case, the data set is not passed.
DSNAME	*dsname(areaname)*	The name is used; areaname is ignored.
	dsname(generation)	The name is used; generation is ignored.
	dsname(member)	The name is used; member is ignored.
	All temporary *dsnames*	Because VSAM data sets are built by Access Method Services, which uses the data-set name supplied in the DEFINE command, temporary names cannot be used with VSAM.
	All backward DD references of the form **.ddname*	If the object referred to is a cluster and the data set and index reside on unlike devices, the results of a backward DD reference are unpredictable.
LABEL	**BLP, NL, NSL**	Because these subparameters have no meaning for direct-access devices, they do not apply for VSAM data sets, which all reside on direct-access storage.
	IN	Because IN is used to override DCB subparameters and the DCB parameter does not apply to VSAM data sets, IN does not apply.
	OUT	Because OUT is used to override DCB subparameters and the DCB parameter does not apply to VSAM data sets, OUT does not apply.
	NOPWREAD	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter.

Figure 14. DD parameters you should avoid with VSAM (Part 1 of 2)

Parameter	Subparameter	Comment
	PASSWORD	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter.
	SL, SUL	Although these parameters apply to direct-access storage devices, SL is always used for VSAM, whether you specify SL, SUL, or neither.
SPACE		VSAM data spaces are initially allocated as a result of the Access Method Services DEFINE command. If SPACE is specified, therefore, an extent is allocated that is never used by VSAM. Moreover, an Access Method Services request for space may fail as a result of the SPACE acquisition of space.
SYSOUT		If SYSOUT is coded with a mutually exclusive parameter (for example, DISP), the job step is terminated with an error message.
UCS	All	Because this parameter applies only to unit-record devices, it does not apply to VSAM.
UNIT	**AFF**	You must use this subparameter carefully. If the cluster components, the data and its index, reside on unlike devices, the results of UNIT=AFF are unpredictable.
VOLUME	**REF**	You must use this subparameter carefully. If the referenced volumes are not a subset of those contained in the catalog record for the data set, the results are unpredictable.
	vol seq number	Results are unpredictable.
	vol count	This subparameter is used to request some number of nonspecific volumes. Because all VSAM volumes must be specifically defined before processing, volcount is not applicable to VSAM data sets.
*		Because there is no way to get VSAM data into the input stream, this parameter has no application with VSAM.

Figure 14. DD parameters you should avoid with VSAM (Part 2 of 2)

Creating and Retrieving Indexed Sequential Data Sets

Indexed sequential (ISAM) data sets are created and retrieved using special subsets of DD statement parameters and subparameters. Each data set can occupy up to three different areas of space:

1. Prime area — This area contains data and related track indexes. It exists for all indexed sequential data sets.
2. Overflow area — This area contains overflow from the prime area when new data is added. It is optional.
3. Index area — This area contains master and cylinder indexes associated with the data set. It exists for any indexed sequential data set that has a prime area occupying more than one cylinder.

Indexed sequential data sets must reside on direct access volumes. The data set can reside on more than one volume and the device types of the volumes may in some cases differ.

Creating an Indexed Sequential Data Set

One to three DD statements can be used to define a new indexed sequential data set. When using three DD statements to define the data set, each DD statement defines a different area and the areas must be defined in the following order:

1. Index area.
2. Prime area.
3. Overflow area.

When using two DD statements to define the data set, the areas must be defined in the following order:

1.	Index area.	1.	Prime area. (optionally, Index area)
	or		
2.	Prime area.	2.	Overflow area.

When using one DD statement to define the data set, you are defining the prime area and, optionally, the index area.

When more than one DD statement is used to define the data set, assign a ddname only to the first DD statement; the name field of the other statements must be blank.

The only DD statement parameters that can be coded when defining a new indexed sequential data set are the DSNAME, UNIT, VOLUME, LABEL, DCB, DISP, and SPACE parameters. When to code each of these parameters and what restrictions apply are described in the following paragraphs.

The DSNAME Parameter

The DSNAME parameter is required on any DD statement that defines a new temporary or nontemporary indexed sequential data set. To identify the area you are defining, you follow the DSNAME parameter with the area: DSNAME=name(INDEX). DSNAME=name(PRIME), or DSNAME=name(OVFLOW). If you are using only one DD statement to define the data set, code DSNAME=name(PRIME) or DSNAME=name.

When reusing previously allocated space to create an ISAM data set, the DSNAME parameter must contain the name of the old data set to be overlaid.

The UNIT Parameter

The UNIT parameter is required on any DD statement that defines a new indexed sequential data set unless VOLUME=REF=reference is coded. You must request a direct access device in the UNIT parameter and must not request DEFER.

If there are separate DD statements defining the prime and index areas, request the same number of direct access devices for the prime area as there are volumes specified in the VOLUME parameter. You request only one direct access volume for an index area and one for an overflow area.

A DD statement for the index area or overflow area can request a device type different than the type requested on the other statements.

Another way to request a device is to code UNIT=AFF=ddname (except for new data sets), where the named DD statement requests the direct access device or device type you want.

The VOLUME Parameter

The VOLUME parameter is required if you want an area of the data set written on a specific volume or the prime area requires the use of more than one volume. (If the prime area and index area are defined on the same statement, you cannot request more than one volume on the DD statement.) Either supply the volume serial number or numbers in the VOLUME parameter or code VOLUME=REF=reference. In all cases, the VOLUME parameter can be used to request a private volume (PRIVATE).

Notes:

1. If a new ISAM data set is being created with a nonspecific volume request and its DSNAME already exists on a volume eligible for allocation, the job might fail due to duplicate names on the volume. If the volume selected for the new data set already contains a data set with the same name, the job fails. If the old data set with a duplicate name resides on another volume than the one selected for the new data set, however, the new data set is not affected and will be added to the volume. You can correct job failures because of duplicate data set names by scratching the old data set or by renaming the new data set before resubmitting the job.

2. If a nonspecific volume request for an ISAM data set is made via a single DD statement and space is not available on the volume selected for allocation, the job fails.

The LABEL Parameter

The LABEL parameter need only be coded to specify a retention period (EXPDT or RETPD) or password protection (PASSWORD).

The DCB Parameter

The DCB parameter must be coded on every DD statement that defines an indexed sequential data set. At minimum, the DCB parameter must contain DSORG=IS or DSORG=ISU. Other DCB subparameters can be coded to complete the data control block if it has not been completed by the processing program. When more than one DD statement is used to define the data set, code all the DCB subparameters on the first DD statement. Code DCB=*.ddname on the remaining statement or statements; ddname is the name of the DD statement that contains the DCB subparameters.

When reusing previously allocated space and recreating an ISAM data set, desired changes in the DCB parameter must be coded on the DD statement. Although you are creating a new data set, some DCB subparameters cannot be changed if you want to use the space the old data set used. The DCB subparameters you can change are: BFALN, BLKSIZE, CYLOFL, DSORG, KEYLEN, LRECL, NCP, NTM, OPTCD, RECFM, and RKP.

The DISP Parameter

If you are creating a new data set and not reusing preallocated space, the DISP parameter need only be coded if you want to keep, DISP=(,KEEP), catalog, DISP=(,CATLG), or pass, DISP=(,PASS), the data set. If reusing previously allocated space and recreating an ISAM data set, code DISP=OLD. The newly created data set will overlay the old one.

In order to catalog the data set when DISP=(,CATLG) is coded or pass the data set when DISP=(,PASS) is coded, the data set must be defined on only one DD statement. If the data set was defined on more than one DD statement and the volumes on which the data set now resides correspond to the same device type, use the Access Method Services DEFINE command to catalog the data set. Refer to the **OS/VS2 Access Method Services** publication for details.

The SPACE Parameter

The SPACE parameter is required on any DD statement that defines a new indexed sequential data set. Use either the recommended nonspecific allocation technique or the more restricted absolute track (ABSTR) technique. If more than one DD statement is used to define the data set, all must request space using the same technique.

Nonspecific Allocation Technique

You must request the primary quantity in cylinders (CYL). When the DD statement that defines the prime area requests more than one volume, each volume is assigned the number of cylinders requested in the SPACE parameter.

One of the subparameters of the SPACE parameter, the "index" subparameter, is used to indicate how many cylinders are required for an index. When one DD statement is used to define the prime and index areas and you want to explicity state the size of the index, code the "index" subparameter.

The CONTIG subparameter can be coded in the SPACE parameter. However, if CONTIG is coded on one of the statements, it must be coded on all of them.

You cannot request a secondary quantity for an indexed sequential data set. Also, you cannot code the subparameters RLSE, MXIG, ALX, and ROUND.

Absolute Track Technique

The number of tracks requested must be equal to one or more whole cylinders. The address of the beginning track must correspond with the first track of a cylinder other than the first cylinder on the volume. When the DD statement that defines the prime area requests more than one volume, space is allocated for the prime area beginning at the specified address and continuing through the volume and onto the next volume until the request is satisfied. (This can only be done if the volume table of contents of the second and all succeeding volumes is contained within the first cylinder of each volume.)

One of the subparameters of the SPACE parameter, the "index" subparameter, is used to indicate how many tracks are required for an index. The number of tracks specified must be equal to one or more cylinders. When one DD statement is used to define the prime and index areas and you want to explicity state the size of the index, code the "index" subparameter.

Note: If the indexed sequential data set is to reside on more than one volume and an error is encountered as the volumes are being allocated to the data set, follow this procedure before resubmitting the job: Use the IEHPROGM utility program to scratch the data set labels on any of the volumes to which the data set was successfully allocated. This utility program is described in the chapter "The IEHPROGM Program" in **OS/VS Utilities**.

Area Arrangement of an Indexed Sequential Data Set

When creating an indexed sequential data set, the arrangement of the areas is based on two criteria:

1. The number of DD statements used to define the data set.
2. What area each DD statement defines.

An additional criterion is used when you do not include a DD statement that defines the index area:

3. Is an index size coded in the SPACE parameter on the DD statement that defines the prime area?

Figure 23 in the "Reference Tables" section illustrates the different arrangements that can result based on the criteria listed above. In addition, it indicates what restrictions apply on the number and types of devices that can be requested.

Retrieving an Indexed Sequential Data Set

If all areas of an existing indexed sequential data set reside on volumes of the same device type, you can retrieve the entire data set with one DD statement. If the index or overflow resides on a volume of a different device type, use two DD statements. If the index and overflow reside on volumes of different device types, use three DD statements to retrieve the data set. The DD statements are coded in the following order:

1. First DD statement - defines the index area
2. Second DD statement - defines the prime area
3. Third DD statement - defines the overflow area

The only DD statement parameters that can be coded when retrieving an indexed sequential data set are the DSNAME, UNIT, VOLUME, DCB, and DISP parameters. When to code each of these parameters and what restrictions apply are described in the following paragraphs.

The DSNAME Parameter

The DSNAME parameter is always required. Identify the data set by its name, however, it is not necessary to include the the terms INDEX, PRIME, or OVFLOW when retrieving an indexed sequential data set. If the data set was passed from a previous step, identify it by a backward reference.

The UNIT Parameter

The UNIT parameter must be coded unless the data set resides on one volume and was passed. You identify in the UNIT parameter the device type and how many of these devices are required.

If the data set resides on more than one volume and the volumes correspond to the same device type, you need only one DD statement to retrieve the data set. Request one device in the UNIT parameter per volume. If the index or overflow area of the data set resides on a different type of volume than the other areas, you must use two DD statements to retrieve the data set. On one DD statement, request the device type required to retrieve the index or overflow area. On the other DD statement, request the device type and the number of devices required to retrieve the prime area and the overflow area if the overflow area resides on the same device type. If the index and the overflow areas reside on different device types from the prime area, a third DD statement is needed.

The VOLUME Parameter

The VOLUME parameter must be coded unless the data set resides on one volume and was passed from a previous step. Identify in the VOLUME parameter the serial numbers of the volumes on which the data set resides. Code the serial numbers in the same order as they were coded on the DD statements used to create the data set.

The DCB Parameter

The DCB parameter must always contain DSORG=IS or DSORG=ISU. You do not have to code other DCB subparameters if the data set is passed from a previous step or is cataloged. However, you can code other DCB subparameters to complete the data control block if it has not been completed by the processing program.

The DISP Parameter

The DISP parameter must always be coded. The first subparameter of the DISP parameter must be SHR or OLD. You can, optionally, assign a disposition as the second subparameter.

Examples of Creating and Retrieving an Indexed Sequential Data Set

The following job creates an indexed sequential data set on one 3330 volume.

```
//ISAMJOB   JOB     ,,MSGLEVEL=(1,1),PERFORM=25
//STEP1     EXEC    PGM=INCLUDE
//DD1       DD      DSN=DATASET1(INDEX),DISP=(NEW,KEEP),UNIT=3330,
//                  VOL=SER=777777,SPACE=(CYL,(10),,CONTIG),
//                  DCB=(DSORG=IS,RECFM=F,LRECL=80,RKP=1,KEYLEN=8)
//          DD      DSN=DATASET1(PRIME),DISP=(NEW,KEEP),UNIT=3330,
//                  VOL=REF=*.DD1,SPACE=(CYL,(25),,CONTIG),DCB=*.DD1
//          DD      DSN=DATASET1(OVFLOW),DISP=(NEW,KEEP),UNIT=3330,
//                  VOL=REF=*.DD1,SPACE=(CYL,(25),,CONTIG),DCB=*.DD1
```

The following job includes the DD statements required to retrieve the indexed sequential data set created above.

```
//RETRISAM  JOB     ,,MSGLEVEL=(1,1),PERFORM=25
//STEP1     EXEC    PGM=RETRIEVE
//DDISAM    DD      DSN=DATASET1,DCB=DSORG=IS,UNIT=3330,DISP=OLD,
//                  VOL=SER=777777
```

The following job creates an indexed sequential data set on one 3330 and two 2314 volumes.

```
//ISAMJOB   JOB     ,,MSGLEVEL=(1,1),PERFORM=25
//STEP1     EXEC    PGM=IEFISAM
//DDISAM    DD      DSN=DATASET2(INDEX),DISP=(NEW,KEEP),UNIT=3330,
//                  VOL=SER=888888,SPACE=(CYL,10,,CONTIG),DCB=(DSORG=IS,
//                  RECFM=F,LRECL=80,RKP=1,KEYLEN=8)
//          DD      DSN=DATASET2(PRIME),DISP=(,KEEP),UNIT=2314,
//                  VOL=SER=999999,SPACE=(CYL,10,,CONTIG),DCB=*.DDISAM
//          DD      DSN=DATASET2(OVFLOW),DISP=(,KEEP),UNIT=2314,
//                  VOL=SER=AAAAAA,SPACE=(CYL,10,,CONTIG),DCB=*.DDISAM
```

The following job includes the DD statements required to retrieve the indexed sequential data set created above.

```
//RERISAM   JOB     ,,MSGLEVEL=(1,1),PERFORM=25
//STEP1     EXEC    PGM=IEFISAM
//DDISAM    DD      DSN=DATASET2,DCB=DSORG=IS,DISP=OLD,UNIT=3330,
//                  VOL=SER=888888
//          DD      DSN=DATASET2,DCB=DSORG=IS,DISP=OLD,UNIT=(2314,2),
//                  VOL=SER=(999999,AAAAAA)
```

Creating and Retrieving Generation Data Sets

A generation data set is one of a collection of successive, historically related, cataloged data sets known as a generation data group. The system keeps track of each data set in a generation data group as it is created so that new data sets can be chronologically ordered and old ones easily retrieved.

To create or retrieve a generation data set, identify the generation data group name in the DSNAME parameter and follow the group name with a relative generation number. When creating a generation data set, the relative generation number tells the system whether this is the first data set being added during the job, the second, the third, etc. When retrieving a generation data set, the relative generation number tells the system how many data sets have been added to the group since this data set was added.

Relative generation numbers are obtained from the catalog as it existed:

- For JES2, at the beginning of the first step that specifies the generation data set by relative generation number.
- For JES3, when the job is setup, and again by the operating system at the beginning of the first step that specifies the generation data set by relative generation number. If the most recent data set is not the same at both times, the results are unpredictable.

A generation data group can consist of cataloged sequential, partitioned, and direct data sets residing on tape volumes, direct access volumes, or both. Generation data sets can have like or unlike DCB attributes and data set organizations. If the attributes and organizations of all generations in a group are identical, the generations can be retrieved together as a single data set (up to 255 data sets can be retrieved in this way). The retrieval order is last in-first out. If the generation data group resides on more than one device type, all generations cannot be retrieved together.

Building a Generation Data Group Base Entry

Before defining the first generation data set, build a generation data group base entry in a VSAM or OS CVOL catalog. This provides for as many generation data sets (up to 255) as you would like to have in the generation data group. The system uses the base to keep track of the chronological order of the generation data sets. Use the Access Method Services DEFINE command to build generation data group bases in a VSAM catalog. This command is described in **OS/VS2 Access Method Services.**

Another requirement of generation data groups is that a data set label list exist. The system uses this label to refer to DCB attributes when you define a new generation data set. There are two ways to satisfy this requirement: (1) create a model data set label on the same volume as the catalog before defining the first generation data set; or (2) use the DCB parameter to refer the system to an existing cataloged data set each time you define a new generation data set.

Creating a Model Data Set Label

To create a model data set label, define a data set and request that it be placed on the same volume as the generation data group base. This ensures that there is always a data set label on the same volume as the catalog to which the system can refer.

The name assigned to the data set can be the same or different than the name assigned to the generation data group. (If you assign the same name for both, the data set associated with the model data set label cannot be cataloged.) Request a space allocation of zero tracks or cylinders. The DCB attributes that can be supplied are DSORG, OPTCD, BLKSIZE, LRECL, KEYLEN, and RKP.

You need not create a model data set label for every generation data group whose indexes reside on the same volume. Instead, create one model data set label to be used by any number of generation data groups. When creating a generation data set, specify the name of the model in the DCB parameter and follow the name with a list of all the DCB subparameters required for the new generation data set that are different than specified in the model; that is, DCB=(dsname,list of attributes).

Referring the System to a Cataloged Data Set

If there is a cataloged data set that resides on the same volume as the generation data group index and you are sure that data set will exist as long as you are adding data sets to the generation data group, you need not create a model data set label. When creating a generation data group, specify the name of the cataloged data set in the DCB parameter by coding DCB=dsname. If all the DCB attributes are not contained in the label of the cataloged data set, or if you want to override certain attributes, follow the data set name with these attributes; that is, DCB=(dsname,list of attributes).

Creating a Generation Data Set

When defining a new generation data set, always code the DSNAME, DISP, and UNIT parameters. Other parameters you might code are the VOLUME, SPACE, LABEL, and DCB parameters.

The DSNAME Parameter

In the DSNAME parameter, code the name of the generation data group followed by a number enclosed in parentheses. This number must be 1 or greater. If this is the first data set you are adding to a particular generation data group during the job, code +1 in parentheses. Each time during the job you add a data set to the same generation data group, increase the number by one. When the first character is a plus (+), the remaining digits (three digits or less) must not exceed 255.

Any time you refer to this data set later in the job, use the same relative generation number as was used earlier. At the end of the job, the system updates the relative generation numbers of all generations in the group to reflect the additions.

Note: Unpredictable results can occur if you use a relative generation number that causes the actual generation number to exceed G9999.

The DISP Parameter

New generations should be assigned a status of NEW and a disposition of CATLG in the DISP parameter; that is, DISP=(NEW,CATLG). If the DISP parameter is not specified, the system assumes DISP=(NEW,DELETE) and the new generation will be deleted at the end of the step.

The UNIT Parameter

The UNIT parameter is required on any DD statement that defines a new generation data set unless VOLUME=REF=reference is coded. In the UNIT parameter, identify the type of devices you want (tape or direct access).

The VOLUME Parameter

You can assign a volume in the VOLUME parameter or let the system assign one for you. The VOLUME parameter can also be used to request a private volume (PRIVATE) and to indicate that more volumes can be required (volume count).

The SPACE Parameter

The SPACE parameter is coded only when the generation data set is to reside on a direct access volume.

The LABEL Parameter

You can specify label type, password protection (PASSWORD), and a retention period (EXPDT or RETPD) in the LABEL parameter. If the data set will reside on a tape volume and is not the first data set on the volume, specify a data set sequence number.

The DCB Parameter

A model data set label that has the same name as the group name may exist. If this is so, and if the label contains all the attributes required to define this generation, you need not code the DCB parameter. If all the attributes are not contained in the label, or if you want to override certain attributes, code DCB=(list of attributes).

If a model data set label has a different name than the group name and if the label contains all the attributes required to define this generation data set, only the name of the data set associated with the model data set label need be coded. Code the name in the DCB parameter, that is, DCB=dsname. If all the attributes are not contained in the label, or if you want to override certain attributes, follow the data set name with these attributes; that is, DCB=(dsname,list of attributes).

If a model data set label does not exist, you must code the name of a cataloged data set that resides on the same volume as the generation data group index. If all the attributes are not contained in the label for this data set, or if you want to override certain attributes, follow the data set name with these attributes.

Retrieving a Generation Data Set

To retrieve a generation data set, always code the DSNAME and DISP parameters. Other parameters you might code are the UNIT, LABEL, and DCB parameters.

The DSNAME Parameter

Using the DSNAME parameter you can retrieve a single generation data set or all of the generation data sets in the generation data group.

Retrieving a Single Generation Data Set If you want to retrieve a single generation data set, code the name of the generation data group followed by a number enclosed in parentheses in the DSNAME parameter. This number can be a maximum of four characters. The number coded depends on which generation data set is to be retrieved. To retrieve the most recent data set, code a zero (0). If the first character is zero (0), the remaining characters must be zero or blanks.

To retrieve data sets created prior to the most recent data set, code a minus value (-nnn). When nnn is a minus value the remaining digits (3 digits or less) must not exceed 255. The value of nnn is determined by the relation of the desired data set to the most current data set. Minus one (-1) refers to the data set created immediately preceding the most recent data set; minus two (-2) refers to the data set created preceding the data set identified by the minus one value.

For example:

WEEKLY.PAYROLL is the name of a generation data group.

```
DSN=WEEKLY.PAYROLL (0)      The most recent generation data set.
DSN=WEEKLY.PAYROLL (-1)     Last week's generation data set.
DSN=WEEKLY.PAYROLL (-2)     Generation data set of two weeks ago.
```

Relative generation numbers are maintained by the system only when generation data sets are specified using relative generation numbers.

Retrieving All Generation Data Sets If you want to retrieve all generations of a generation data group as a single data set, specify the generation data group name without a generation number in the DSNAME parameter: for example, DSNAME=WEEKLY.PAYROLL where WEEKLY.PAYROLL is the generation data group name.

You can retrieve all generations in this manner only if the DCB attributes and data set organization of all generations are identical.

When you specify the generation data group name without a generation number, the operating system treats your request as a concatenation of all existing data sets in the generation data group, starting with the most recent data set and ending with the oldest data set. In addition, all data sets, except the most recent, will have unit affinity to the most recent data set. For further information see "Concatenating Data Sets" in this publication.

The relative generation number of the most recent data set is obtained from the catalog as it existed:

- For JES2, at the beginning of the first step that specifies the generation data set by relative generation number.
- For JES3, when the job is setup, and again by the operating system at the beginning of the first step that specifies the generation data set by relative generation number. If the most recent data set is not the same at both times, the results are unpredictable.

The DISP Parameter

The DISP parameter must always be coded. The first subparameter of the DISP parameter must be OLD, SHR, or MOD. You can, optionally, assign a disposition as the second subparameter. The second subparameter must be specified for a generation data group. You should avoid coding PASS as the second subparameter when you retrieve all generations of a generation data group as a single data set. In all such retrievals the unit and volume information for each generation level is obtained from the catalog, and not from the pass mechanism. If you code the DISP subparameter MOD for a generation data set and the specified relative generation does not exist in the catalog, the operating system changes the disposition to NEW.

The UNIT Parameter

Code the UNIT parameter when you want more than one device assigned to the data set. Code the number of devices you want in the unit count subparameter, or, if the data set resides on more than one volume and you want as many devices as there are volumes, code P in place of the unit count subparameter.

The VOLUME Parameter

Use the VOLUME parameter to request a private volume (PRIVATE) and to indicate that more volumes might be required (volume count). A volume serial number specified for an old generation data group is ignored; the system obtains the volume serial number from the catalog.

The LABEL Parameter

Code the LABEL parameter when the data set resides on tape and has other than standard labels. If the data set is not the first data set on the volume, specify the data set sequence number. If the data set sequence number is coded for a GDG ALL request, it is ignored; the data set sequence number will be taken from the catalog.

The DCB Parameter

Code DCB=(list of attributes) when the data set has other than standard labels and DCB information is required to complete the data control block. Do not code DCB=dsname when retrieving a generation data set.

Submitting a Job for Restart

Certain rules apply when you refer to generation data sets in a job submitted for restart (the RESTART parameter is coded on the JOB statement).

If you have installed MVS/System Product Release 2 (5740-XYN or 5740-XYS) without installing Data Facility/Device Support Release 1 Enhancements (5740-AM7), do not use the checkpoint/restart facility.

If you have installed Data Facility/Device Support with either of the MVS/System Products, you can use the checkpoint/restart facility with certain restrictions.

For additional information concerning these restrictions, see **OS/VS2 MVS Checkpoint/Restart** and **OS/VS2 MVS System Programming Library: Job Management.**

For step restart: If step restart is performed, generation data sets that were created and cataloged in steps preceding the restart step must not be referred to in the restart step or in steps following the restart step by means of the same relative generation numbers that were used to create them. Instead, you must refer to a generation data set by means of its present relative generation number. For example, if the last generation data set created and cataloged

was assigned a generation number of +2, it would be referred to as 0 in the restart step and in steps following the restart step. In this case, the generation data set assigned number of +1 would be referred to as -1.

For checkpoint restart: If generation data sets created in the restart step were kept instead of cataloged, that is, DISP=(NEW,CATLG,KEEP) was coded, you can, during checkpoint restart, refer to these data sets and generation data sets created and cataloged in steps preceding the restart step by means of the same relative generation numbers that were used to create them.

Example of Creating and Retrieving Generation Data Sets

The following job step includes the DD statements that could be used to add three data sets to a generation data group.

```
//STEPA     EXEC    PGM=PROCESS
//DD1       DD      DSNAME=A.B.C(+1),DISP=(NEW,CATLG),UNIT=2400,
//                  VOL=SER=13846,LABEL=(,SUL)
//DD2       DD      DSNAME=A.B.C(+2),DISP=(NEW,CATLG),UNIT=3330,
//                  VOL=SER=10311,SPACE=(480,(150,20))
//DD3       DD      DSNAME=A.B.C(+3),DISP=(NEW,CATLG),UNIT=2314,
//                  VOL=SER=28929,SPACE=(480,(150,20)),
//                  DCB=(LRECL=120,BLKSIZE=480)
```

The first two DD statements do not include the DCB parameter because a model data set label exists on the same volume as the generation data group index and has the same name as the generation data group (A.B.C). Since the DCB parameter is coded on the third DD statement, the attributes LRECL and BLKSIZE, along with the attributes included in the model data set label, are used.

The following job includes the DD statements required to retrieve the generation data sets defined above when no other data sets have been added to the generation data group.

```
//JWC       JOB     CLASS=B
//STEP1     EXEC    PGM=REPORT9
//DDA       DD      DSNAME=A.B.C(-2),DISP=OLD,LABEL=(,SUL)
//DDB       DD      DSNAME=A.B.C(-1),DISP=OLD
//DDC       DD      DSNAME=A.B.C(0),DISP=OLD
```

Creating and Using a Subsystem Data Set

Use the DD SUBSYS parameter to:

- Specify the name of the subsystem that will process the associated subsystem data set
- Specify up to 254 subsystem-defined parameters that describe the subsystem data set to the subsystem

The subsystem processes the subsystem-defined parameters according to its own rules.

The SUBSYS parameter can only be used if Subsystem Support (SU29) has been installed.

When you specify the SUBSYS parameter, the significance of certain DD statement parameters may be altered by the subsystem. To determine if a particular subsystem alters the significance of any DD statement parameters, and if it does, to determine which statements are affected and how they are affected, refer to the documentation for the subsystem.

If you specify the DUMMY parameter, MVS invokes the specified subsystem to syntax check the subsystem-defined parameters. If the syntax is acceptable, MVS assigns a dummy status to the data set and processes the request as a dummy request.

If you request UNIT affinity to a subsystem data set, MVS substitutes SYSALLDA as the UNIT parameter specification.

Cataloged and In-Stream Procedures

Applications that require many control statements and are used on a regular basis can be considerably simplified through the use of cataloged and in-stream procedures. A cataloged procedure is a set of job control statements that are placed in a partitioned data set known as the procedure library; an in-stream procedure is a set of job control statements that are placed in the input stream within a job. You can execute a procedure simply by specifying its name on an EXEC statement in your job. This section describes how to write and use cataloged and in-stream procedures.

This section includes three topics:

- Writing Cataloged and In-Stream Procedures
- Using Cataloged and In-Stream Procedures
- Using Symbolic Parameters

Writing Cataloged and In-Stream Procedures

Cataloged and in-stream procedures are simply the job control statements needed to perform an application. A procedure contains one or more procedure steps, each step consisting of an EXEC statement that identifies the program to be executed and DD statements defining the data sets to be used or produced by the program. The program requested on the EXEC statement must exist in a private or the system library. If you do request a program that is contained in a private library, the procedure step calling that program must include a DD statement with the ddname STEPLIB that defines the private library; the STEPLIB DD statement is described in the chapter, "Creating and Using Private and Temporary Libraries."

Cataloged and in-stream procedures cannot contain:

- EXEC statements that refer to other cataloged or in-stream procedures;
- JOB, delimiter, or null statements;
- DD statements defining private libraries to be used throughout the job (DD statements with the ddname JOBLIB);
- DD statements defining data in the input stream (statements including the * or DATA parameters).
- Any JES2 control statements; they are ignored.
- Any JES3 control statements; they are ignored in cataloged procedures only.

Identifying an In-Stream Procedure

To identify an in-stream procedure, code the PROC and PEND job control statements.

On the PROC statement, which must be the first statement in an in-stream procedure, assign the procedure a name. This name is the name that a programmer codes to call the procedure. Optionally, you can also assign default values to symbolic parameters contained in the procedure and code comments. (A symbolic parameter is a symbol preceded by an ampersand that stands for a parameter, a subparameter, or a value in a procedure; including symbolic parameters in a procedure is described in detail in the chapter "Using Symbolic Parameters.") If you do not assign default values to symbolic parameters on the PROC statement, you cannot code comments. The simplest form of the PROC statement, to identify an in-stream procedure named PAYROLL, would be:

```
//PAYROLL   PROC
```

The PEND statement marks the end of the in-stream procedure. You can include a name on the PEND statement and comments, but these are optional. Both of the following examples are acceptable:

```
//ENDPROC   PEND    end of in-stream procedure
//          PEND
```

The following example illustrates an in-stream procedure named SALES consisting of two procedure steps. Note that STEP2 includes a STEPLIB DD statement to define the private library in which the program JUGGLE can be found.

```
//SALES     PROC
//STEP1     EXEC    PGM=FETCH
//DD1A      DD      DSNAME=RECORDS(BRANCHES),DISP=OLD
//DD1B      DD      DSNAME=RECORDS(MORGUE),DISP=MOD
//STEP2     EXEC    PGM=JUGGLE
//STEPLIB   DD      DSNAME=PRIV.WORK,DISP=OLD
//DD2A      DD      SYSOUT=A
//          PEND
```

Placing a Cataloged Procedure in a Procedure Library

The major difference between cataloged and in-stream procedures is where they are placed. Cataloged procedures must be placed in a procedure library before being used. In-stream procedures are placed within the job that calls them. A procedure library is simply a partitioned data set containing cataloged procedures. IBM supplies a procedure library named SYS1.PROCLIB, but the installation can have additional procedure libraries with different names. When a programmer calls a cataloged procedure, he receives a copy of the procedure; therefore, a cataloged procedure can be used by more than one programmer simultaneously.

To add a procedure to a procedure library, use the IEBUPDTE utility program. You can also use the IEBUPDTE utility to permanently modify an existing procedure. (Before modifying an existing cataloged procedure, however, you must notify t he operator; he must delay the execution of jobs that might use the procedure library while it is being updated.) Details on using the IEBUPDTE utility are included in **OS/VS Utilities**. In JES3, you can use the procedure library update feature to modify an existing procedure. The UPDATE parameter on the JES3 MAIN statement indicates that a procedure library is being updated and causes all jobs using the library to be held until the update is complete. Before placing or modifying a cataloged procedure in a procedure library, test it without overriding any parameters to ensure that the procedure statements are syntactically correct. Additionally, test the procedure by first running it as an in-stream procedure. In-stream testing enables you to detect any errors in overridden parameters prior to cataloging the procedure.

No special job control statements are used to identify a cataloged procedure. The PEND statement is never used and the PROC statement is optional. You need code the PROC statement as the first statement in a cataloged procedure only when you want to assign default values to symbolic parameters. The name of the PROC statement is not necessarily the name of the cataloged procedure; you assign the procedure a name when adding it to the procedure library.

Allowing for Changes in Cataloged and In-Stream Procedures

The usefulness of cataloged and in-stream procedures is destroyed if a programmer who uses the procedure has to permanently modify the procedure every time he wants to make a change. When writing a procedure, you can define, as symbolic parameters, those parameters, subparameters and values that are likely to vary each time the procedure is used. For details on coding symbolic parameters, see the chapter "Using Symbolic Parameters."

Using Cataloged and In-Stream Procedures

To use a cataloged or in-stream procedure, specify the procedure name on an EXEC statement. You can modify the procedure by adding DD statements, overriding, adding, or nullifying parameters on EXEC and DD statements, and assigning values to symbolic parameters. Calling and modifying procedures is explained in greater detail in the following paragraphs.

How to Call Cataloged and In-Stream Procedures

To call a cataloged or in-stream procedure, you identify the procedure on the EXEC statement of the step calling the procedure by coding as the first operand on the EXEC statement.

- The procedure name.
- PROC= the procedure name.

A cataloged procedure must exist in the procedure library before you attempt to use it. JES2 or JES3 is responsible for fetching cataloged procedures. Refer to "Routing a Job Through the System" to see how JES2 or JES3 determines what library to select. When using an in-stream procedure, include the procedure, beginning with a PROC statement and ending with a PEND statement, with the job control language for the job; the procedure must follow the JOB statement but appear before the EXEC statement that calls it. You can include as many as fifteen uniquely named in-stream procedures in one job and can use each procedure as many times as you wish in the job.

To call a cataloged procedure named PROCESSA, you would code:

```
//CALL      EXEC    PROCESSA          or
//CALL      EXEC    PROC=PROCESSA
```

On the EXEC statement, you can also code changes you would like to make for this execution of the procedure.

Modifying Cataloged and In-Stream Procedures

You can modify a procedure by:

- Assigning values to or nullifying symbolic parameters contained in the procedure.
- Overriding, adding, or nullifying parameters on EXEC and DD statements in the procedure.
- Adding DD statements to the procedure.

All changes you make are in effect only during the current execution of the procedure. For a discussion of symbolic parameters, see the chapter "Using Symbolic Parameters." Other modifications are described in the following sections.

Modifying Parameters on an EXEC Statement

To override, add, or nullify a parameter on an EXEC statement in a procedure, identify on the EXEC statement that calls the procedure the parameter you are changing, the name of the EXEC statement on which the parameter appears, and the change to be made:

```
//CALL      EXEC    procedurename,parameter.procstepname=value
```

When overriding a parameter, the value coded for the parameter on the EXEC statement calling the procedure replaces the value assigned in the procedure. When adding a parameter, that parameter is used in the execution of the procedure step. When nullifying a parameter, you do not follow the equal sign with a value; the value assigned to the parameter in the procedure is ignored. All changes made are in effect only for the current execution of the procedure.

You can make more than one change to each EXEC statement in the procedure, and you can change parameters on more than one EXEC statement in the procedure. You cannot, however, change the PGM parameter. When making changes on different steps in the procedure, code all changes for one procedure step before changes to a subsequent step.

For example, the first three EXEC statements in a procedure named IRISH are:

```
//STEP1     EXEC    PGM=YEATS,PARM='*14863'
//STEP2     EXEC    PGM=NOLAN
//STEP3     EXEC    PGM=SYNGE,TIME=(2,30)
```

and you want to make the following changes:

- Nullify the PARM parameter in STEP1.
- Add the COND parameter, specifying the test (8,LT), in STEP2.
- Change the time limit in the TIME parameter in STEP3 to 4 minutes.

On the EXEC statement calling the procedure, you would code:

```
//CALL      EXEC    IRISH,PARM.STEP1=,
//                  COND.STEP2=(8,LT),TIME.STEP3=4
```

In the above example, code TIME.STEP3=1440 to nullify the TIME parameter. If you code TIME.STEP3=, the default time value for the job class is assigned.

You can omit naming the procedure step when changing a parameter. When you do this, the procedure is modified as follows:

- If the PARM parameter is coded, it applies only to the first procedure step. If a PARM parameter appears in a later EXEC statement in the called procedure, it is nullified.
- If the TIME parameter is coded, it applies to the total procedure. If the TIME parameter appears on any of the EXEC statements in the called procedure, it is nullified.
- If any other parameter is coded, it applies to every step in the called procedure. Nullifying the parameter on the EXEC statement calling the procedure causes the parameter to be ignored on every EXEC statement in the procedure; if you assign a value to the parameter on the EXEC statement calling the procedure, the parameter is overridden where it appears in the procedure and added to EXEC statements in the procedure on which it does not appear.

For example, assume the EXEC statements in the procedure named COMPUTE are:

```
//STEP1     EXEC    PGM=LIST,TIME=(1,30)
//STEP2     EXEC    PGM=UPDATE,RD=NC,TIME=2
//STEP3     EXEC    PGM=CHECK,RD=RNC,COND=ONLY
```

You want to make the following changes:

1. Assign a time limit of 4 minutes to the entire procedure; TIME parameters on individual EXEC statements in the procedure will be nullified.
2. Allow automatic step restart for each step of the job by coding RD=R. The RD parameter will be added to the first step of the job and will override the RD parameters in STEP2 and STEP3.

To call the procedure and make these changes, you would code:

```
//CALL      EXEC    COMPUTE,TIME=4,RD=R
```

During the processing of the JCL statements for the job, the EXEC statements appear as:

```
//STEP1     EXEC    PGM=LIST,RD=R
//STEP2     EXEC    PGM=UPDATE,RD=R
//STEP3     EXEC    PGM=CHECK,RD=R,COND=ONLY
```

Any parameter changes that affect every step of the job (by omitting the procedure step name) must be coded on the EXEC statement calling the procedure before changes to parameters on different steps (that is, you include the procedure step name). Time will be a total of four minutes, each step using the remaining amount of time available from the total. If more than four minutes is required, the step will abnormally terminate.

Modifying Parameters on a DD Statement

To override, add, and nullify parameters on a DD statement in a procedure, you include a DD statement containing the changes you want to make after the EXEC statement that calls the procedure. The name of the DD statement containing the changes is composed of the procedure step name and the ddname of the DD statement in the procedure:

```
//procstepname.ddname   DD   parameter=value
```

When overriding a parameter, the value you code replaces the value assigned to the parameter in the procedure. When adding a parameter, the parameter is added to the DD statement in the procedure for the current execution of the procedure. When nullifying a parameter, you do not follow the equal sign with a value; that parameter in the procedure is ignored. Do not nullify a parameter when you are replacing it with a mutually exclusive parameter; it will be nullified automatically. (See Figure 24 for a table of mutually exclusive parameters on the DD statement.) All changes you make are in effect only for the current execution of the procedure. The checking for mutually exclusive parameters on a DD statement is not performed if the DD statement is being overridden. Therefore, all procedures should be executed once without any overriding statements to ensure that they do not contain any mutually exclusive parameters.

You can change more than one parameter on a DD statement and you can change parameters on more than one DD statement in the procedure. However, the DD statements containing the changes must be coded in the same order as the corresponding DD statements in the procedure. Test all new procedures without overriding any parameters to ensure that the procedure statements are syntactically correct.

For example, the first two steps of the cataloged procedure TEA are:

```
//STEP1     EXEC    PGM=SUGAR
//DD1A      DD      DSNAME=DRINK,DISP=(NEW,DELETE),
//                  UNIT=2400,VOL=SER=568998
//DD1B      DD      UNIT=SYSSQ
//STEP2     EXEC    PGM=LEMON
//DD2A      DD      UNIT=2314,DISP=(,PASS),
//                  SPACE=(TRK,(20,2))
```

You want to make the following changes for this invocation:

1. Change the disposition on the DD statement named DD1A to CATLG.
2. Change the volume serial number on the DD statement named DD1A to a non-specific request allowing the operating system to choose the volume.
3. Change the unit on the DD statement named DD1B to TAPE.
4. Change the SPACE parameter on the DD statement named DD2A to SPACE=(CYL,(4,1)).

When calling the procedure, you would code:

```
//CALL          EXEC    TEA
//STEP1.DD1A    DD      DISP=(NEW,CATLG),VOL=SER=
//STEP1.DD1B    DD      UNIT=TAPE
//STEP2.DD2A    DD      SPACE=(CYL,(4,1))
```

When changing DCB keyword subparameters, you need code only those subparameters you are changing. The DCB keyword subparameters you do not code (and for which you do not code a mutually exclusive subparameter) remain unchanged. For example, a DD statement named DD1 in a procedure step named STEP1 contains DCB=(BUFNO=1,BLKSIZE=800,RECFM=FM,BUFL=800). To change the block size to 320 and the buffer length to 320, you would code:

```
//STEP1.DD1      DD      DCB=(BLKSIZE=320,BUFL=320)
```

The subparameters BUFNO and RECFM remain unchanged.

If a DCB positional subparameter is needed, the DCB positional subparameter must be coded on the override statement regardless of whether one exists in the statement to be overridden. To nullify a DCB positional parameter, do not code the DCB positional parameter on the override statement.

For example, a DD statement named DD2 in a procedure step named STEP2 contains DCB=(DSN1,BLKSIZE=80). To change the block size to 400, and copy other DCB information from the cataloged data set named DSN1, you would code:

```
//STEP2.DD1 DD      DCB=(DSN1,BLKSIZE=400)
```

To nullify the DCB parameter, you must nullify each subparameter. For example, if a DD statement in a procedure contains DCB=(RECFM=FB,BLKSIZE=160,LRECL=80), you must code DCB=(RECFM=,BLKSIZE=,LRECL=) in order to nullify the DCB parameter.

To nullify the DUMMY parameter, code the DSNAME parameter on the overriding DD statement and assign a data set name other than NULLFILE. To nullify all the parameters on a DD statement other than DCB, code DUMMY on the overriding DD statement. If you code DUMMY on a DD statement, all other parameters on the DD statement, with the exception of the DCB parameter, are ignored. However, all parameters are syntax checked so they must be correct. (The DUMMY parameter is described in detail in the chapter, "Defining a Dummy Data Set.")

Modifying Parameters on DD Statements that Define Concatenated Data Sets

When a concatenation of data sets is defined in a cataloged procedure and you attempt to override the concatenation with one DD statement, only the first (named) DD statement is overridden. To override others, you must include an overriding DD statement for each DD statement; the DD statements in the input stream must be in the same order as the DD statements in the procedure. The second and subsequent overriding statements must not be named. If you do not wish to change one of the concatenated DD statements, leave the operand field blank on the corresponding DD statement in the input stream. (This is the only case where a blank operand field for a DD statement is valid.)

For example, suppose you are calling a procedure that includes the following sequence of DD statements in STEPC:

```
//DD4        DD      DSNAME=A.B.C,DISP=OLD
//           DD      DSNAME=STRP,DISP=OLD,UNIT=2314,VOL=SER=X12182
//           DD      DSNAME=TYPE3,DISP=OLD,UNIT=2314,VOLUME=SER=BL1421
//           DD      DSNAME=A.B.D,DISP=OLD
```

To override the DD statements that define the data sets named STRP and A.B.D, you would code:

```
//STEPC.DD4 DD
//          DD       DSNAME=INV.CLS,DISP=OLD
//          DD
//          DD       DSNAME=PAL8,DISP=OLD,UNIT=2314,VOL=SER=125688
```

Adding DD Statements to a Procedure

You can add DD statements to a procedure when calling the procedure. These additional DD statements are in effect only during the current execution of the procedure.

To add a DD statement to a procedure step, follow the EXEC statement that calls the procedure and any overriding DD statements for that step with the additional DD statement. The ddname of the DD statement identifies the procedure step to which this statement is to be added and must be assigned a name that is different from all the ddnames in the procedure step. If you do not identify the procedure step in the ddname, the DD statement is added to the step specified by the last DD statement that contains a stepname and modifies a DD in the procedure. If there are no DD statements that contain stepname.ddname, then the DD statement is added to the first step of the procedure.

For example, if you use the following procedure:

```
//LINKS1        PROC
//LK1           EXEC    PGM=IEWL,REGION=512K
//SYSUT1        DD      SPACE=(CYL,(5,2)),UNIT=SYSDA
//SYSPRINT      DD      SYSOUT=A
//SYSLMOD       DD      DISP=OLD,UNIT=3330,VOL=SER=&SER,
//                      DSN=SYS1.TESTLIB
//LK2           EXEC    PGM=IEWL,REGION=512K
//SYSUT1        DD      SPACE=(CYL,(5,2)),UNIT=SYSDA
//SYSPRINT      DD      SYSOUT=A
//SYSLMOD       DD      UNIT=3330,VOL=SER=TSTVOL,
//                      DSN=COPY.TESTLIB,DISP=OLD
```

and you specify these DD statements:

```
//LK2.SYSLMOD   DD      DSN=COPY2.TESTLIB,UNIT=2314,
//                      VOL=SER=ATEST,DISP=SHR
//ADD           DD      DSN=SYS1.LPALIB,DISP=SHR
```

then the DD statement with the ddname of ADD would be added to step LK2. If the DD statement LK2.SYSLMOD had not been coded, then the DD statement ADD would be added to step LK1.

When adding DD statements to a procedure that contains concatenated DD statements, follow the rules outlined in the previous topic "Modifying Parameters on DD Statements That Define Concatenated Data Sets".

Identifying Procedure Statements on an Output Listing

You can request that cataloged and in-stream procedure statements be included on the output listing by coding 1 as the first subparameter in the MSGLEVEL parameter on the JOB statement. (For a description of the MSGLEVEL parameter, see "Requesting Listings of JCL Statements and System Messages.")

Procedure statements are identified on the output listing as illustrated in Figures 15 and 16. The output listing will also show the symbolic parameters and the values assigned to them.

Columns 1,2,3	
XX	cataloged procedure statement you did not override
X/	cataloged procedure statement you did override
XX*	cataloged procedure statement, other than a comment statement, that the system considers to contain only comments
***	comment statement, JES2, and JES3 statements
Note: The X/ identifier applies only to DD statements.	

Figure 15. Identification of Cataloged Procedure Statements on the Output Listing

Columns 1,2,3	
++	in-stream procedure statement you did not override
+/	in-stream procedure statement you did override
++*	in-stream procedure statement, other than a comment statement, that the system considers to contain only comments
***	comment statement, JES2, and JES3 statements
Note: The +/ identifier applies only to DD statements.	

Figure 16. Identification of In-stream Procedure Statements on the Output Listing

Using Symbolic Parameters

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

```
//STEP1     EXEC    PGM=UPDATE,ACCT=(PGMG,&DEPT)
//DD1       DD      DSNAME=INIT,UNIT=&DEVICE,SPACE=(CYL,(&SPACE,10))
//DD2       DD      DSNAME=CHNG,UNIT=2400,DCB=BLKSIZE=&LENGTH
```

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

Defining Symbolic Parameters When Writing a Procedure

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

The symbolic parameter itself is one to seven alphameric and national (#,@,$) characters preceded by a single ampersand. The first character must be alphabetic or national. Since a single ampersand defines a symbolic parameter, you code double ampersands when not defining a symbolic parameter. For example, if you want to pass 543&LEV to a processing program by means of the PARM parameter, you must code PARM='543&&LEV'. The system

treats the double ampersand as if a single ampersand had been coded, and appears in the results.

If double ampersands (& &) or apostrophes (") are specified on the same statement with a symbolic parameter, the symbolic parameter must precede the double ampersands or apostrophes in order for corect substitution to occur.

Keyword parameters that can be coded on the EXEC statement (such as ACCT, COND, and PARM) cannot be used as the name of a symbolic parameter. For example, you cannot code &REGION=200K or REGION=&REGION on the EXEC statement, but you can code REGION=&SIZE.

The definitions used to signify symbolic parameters should be consistent in all the cataloged and in-stream procedures at an installation. For example, every time the programmer is to assign his department number to a symbolic parameter, no matter which procedure he is calling, the symbolic parameter could be defined as &DEPT. In different procedures, you could code ACCT=(43877,&DEPT) and DSNAME=LIBRARY.&DEPT.TALLY. The programmer would assign his department number to the symbolic parameter wherever that symbolic parameter appears in a procedure.

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

Assigning Default Values to Symbolic Parameters

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

Assigning Values to and Nullifying Symbolic Parameters

When a procedure containing symbolic parameters is used, each symbolic parameter must either be assigned a value or nullified. Symbolic parameters are assigned values or nullified in one of two ways:

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

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

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

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

Assigning a Value to a Symbolic Parameter

To assign a value to symbolic parameter, you code:

```
symbolic parameter=value
```

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

There are some rules for assigning values to symbolic parameters:

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

    ```
    //INIT  EXEC    SOMEPROC,PARM='&PRM'
    ```

    The length of the value assigned to &PRM must be less than 107 because the length of the operand field (SOMEPROC,PARM=) is 14.
- If the value contains special characters, enclose the value in apostrophes (the enclosing apostrophes are not considered part of the value). If the special characters include apostrophes, each apostrophe must be shown as two consecutive apostrophes.
- If more than one value is assigned to a symbolic parameter as a default on the PROC statement, only the first value encountered is used; likewise, if more than one value is assigned to a symbolic parameter on an EXEC statement, only the first value encountered is used.
- If a symbolic parameter is a positional parameter followed by other parameters in the statement, it should be followed in the procedure by a period instead of a comma; for example:

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

- A value of literal blanks, that is, VALUE=' ', should not be used to nullify a symbolic parameter.

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

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

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

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

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

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

Nullifying a Symbolic Parameter

To nullify a symbolic parameter, code:

```
symbolic parameter=
```

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

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

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

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

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

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

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

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

Caution Concerning Leading and Trailing Commas

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

For example, you code for a unit request:

```
UNIT=(2314,&MORE,DEFER)
```

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

```
UNIT=(2314,,DEFER)
```

However, if you code:

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

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

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

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

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

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

When a value is assigned to the symbolic parameter, a comma must be included in the value, that is SERNO=',222222'. (Since the comma is a special character, the value is enclosed in single apostrophes.)

When a symbolic parameter precedes information that does not vary, a period may be required after the symbolic parameter to distinguish the end of the symbolic parameter from the beginning of the information that does not vary. A period is required after the symbolic parameter when the character following the symbolic parameter is:

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

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

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

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

If &POSPARM is nullified, the statement appears as:

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

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

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

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

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

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

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

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

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

Example of an In-Stream Procedure Containing Symbolic Parameters

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

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

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

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

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

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

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

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

```
//STEPX     EXEC    TESTPROC,A=IEFBR14,B=,C=2314,D=,E=
```

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

```
//STEP      EXEC    PGM=IEFBR14
//DD1       DD      DSN=,UNIT=2314,VOL=SER=,DISP=,SPACE=(TRK,(10,10,1))
```

There are certain rules common to all parameters. Syntax rules define how to code each parameter; that is, what is required or optional for the specific purpose or process you are requesting. Certain fields on the control statements are common to most parameters, such as, the name field, the operation field, and the operand field. Special characters can be coded on the parameters if you follow guidelines established in the rules for coding.

JCL rules and guidelines are to be followed in coding JCL statements in order to achieve specific results. Non-compliance with the rules and guidelines might result in error messages. Also, non-compliance might produce unpredictable results. The use of statements or operands to achieve results other than those stated in this publication is not supported.

Notation for Defining Control Statement Parameters

The formats of the parameters described in this publication for the JOB, EXEC, DD, JES2, and JES3 statements appear at the beginning of the chapter on the corresponding parameter. Notations used in the format descriptions are described below.

1. Uppercase letters and words are coded on the control statement exactly as they appear in the format description, as well as the following characters:

ampersand	&
asterisk	*
comma	,
equal sign	=
parenthesis	()
period	.

2. Lowercase letters, words, and symbols appearing in the format description represent variables for which specific information is substituted when the parameter is coded.

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

3. Braces {} are a special notation and are never coded on a control statement. Braces are used to group related items; they indicate that you must code one of the items.

   For example, $\left\{\begin{array}{l}\text{TRK}\\\text{CYL}\\\text{block size}\end{array}\right\}$ is part of the format description for the SPACE parameter.

   When coding the SPACE parameter, you must code either TRK, CYL, or a substitute for "block size", which would be a number.

4. Brackets [] are a special notation and are never coded on a control statement. Brackets indicate that the enclosed item or items are optional and you can code one or none of the items.

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

   An example of more than one item enclosed in brackets is

   $$\left[\begin{array}{l}\text{EXPDT=yyddd}\\\text{RETPD=nnnn}\end{array}\right]$$

   which is part of the format description for the LABEL parameter. When coding the LABEL parameter, you can include either EXPDT=yyddd or RETPD=nnnn in the LABEL parameter or omit both.

Sometimes, one of a group of items enclosed in brackets is a comma. Code the comma when none of the other items in the group is used and a following part of the parameter is still to be coded.

For example, [,progname] [,form name]
[,]

is part of the format description for the SYSOUT parameter. When you code the SYSOUT parameter, you have the option of coding both "progname" and ",form name", omitting both, or coding only one. The comma enclosed in brackets with ",progname" must be coded when ",progname" is not coded but ",form name" is coded; that is, you would code: ,,form name.

5. An ellipsis...(three consecutive periods) is a special notation and is never coded on a control statement. An ellipsis is used to indicate that the preceding item can be coded more than once in succession.

   For example, COND=((code,operator),...) is the format description for the COND parameter on the JOB statement. The ellipsis indicates that (code,operator) can be repeated.

Note: Two consecutive periods (..) are used to concatenate symbolic parameters to other information. For example, &DEPT..MACS is the symbolic parameter. If DEPT=D58, then the actual value is D58.MACS.

Fields in JCL Control Statements

Every control statement is logically divided into different fields. There are four fields — name field, operation field, operand field, comments field — but not all of the control statements can contain all of these fields. Figure 17 shows the fields for each statement.

Statement	Columns 1 and 2	Fields
JOB	//	name operation (JOB) operand[1] comments[2]
EXEC	//	name[1] operation (EXEC) operand comments[2]
DD	//	name[1] operation (DD) operand comments[2]
PROC (cataloged)	//	name[1] operation (PROC) operand comments[2]
PROC (in-stream)	//	name operation (PROC) operand[1] comments[2]
PEND	//	name[1] operation (PEND) comments[1]
Command	//	operation (command) operand[1] comments[1]
Delimiter	/*	comments[1]
Null	//	
Statement	**Columns 1, 2, 3**	**Field**
Comment	//*	comments

[1] Optional
[2] Optional -- If operand(s) are not coded, comments cannot be coded.
If operand(s) are coded, comments are optional.

Figure 17. JCL Control Statement Fields

The **name field** identifies the control statement so that other statements and system control blocks can refer to it. The name field is 1 to 8 alphameric and national (#,@,$) characters; the first character must be alphabetic or national. The name field must begin in column 3.

The **operation field** specifies the type of control statement, or, in the case of the command

statement, the command. The operation field must follow the name field and must be preceded and followed by at least one blank.

The **operand field** contains parameters separated by commas. The operand field must follow the operation field and must be preceded and followed by at least one blank. The operand field is described in more detail in the next topic "Parameters in the Operand Field."

The **comments field** contains any information deemed helpful by the person who codes the control statement. The comments field must follow the operand field and must be preceded by at least one blank. The operand field can be continued; it does not have to be completed before you add the comments field.

Control statement fields — except the name field, which must begin in column 3 — can be coded in free form. Free form means that the fields need not begin in a particular column. Separate each field with a blank; the blank serves as a delimiter between fields.

Except for the comment statement, which can be coded through column 80, fields cannot be coded past column 71. If the total length of the fields will exceed 71 columns, you must continue the fields onto one or more succeeding statements. How to continue fields is described under "Continuing Control Statements."

Parameters in the Operand Field

The operand field is made up of two types of parameters: one type is characterized by its position in the operand field in relation to other parameters (a positional parameter); the other type is positionally independent with respect to others of its type, and is characterized by a keyword followed by an equal sign and variable information (a keyword parameter). Both positional parameters and the variable information associated with keyword parameters can assume the form of a list of several items (subparameters) of information.

All positional and keyword parameters and subparameters coded in the operand field must be separated from one another by commas.

Positional parameters must be coded first in the operand field in a specific order. The absence of a positional parameter is indicated by a comma coded in its place. However, if the absent parameter is the last one, or if all later positional parameters are also absent, you need not code replacing commas. If all positional parameters are absent from the operand field, you need not code any replacing commas.

Keyword parameters can be used anywhere in the operand field with respect to one another. Because of this positional independence, you need not indicate the absence of a keyword parameter.

A positional parameter or the variable information in a keyword parameter sometimes assumes the form of a **list of subparameters**. Such a list may be composed of both positional and keyword subparameters that follow the same rules and restrictions as positional and keyword parameters. You must enclose a subparameter list in parentheses, unless the list reduces to either a single keyword subparameter, or a single positional subparameter with no preceding subparameters omitted.

The EXEC statements and DD statements in cataloged and in-stream procedures can contain one other type of parameter — a **symbolic parameter**. A symbolic parameter is characterized by a name preceded by an ampersand (&); a symbolic parameter stands as a symbol for a parameter, a subparameter, or a value. Symbolic parameters allow you to make any information in the operand field of a procedure EXEC statement or DD statement variable. A value to be assumed by a symbolic parameter may be coded on the EXEC statement that calls the procedure. This value is in effect only while the procedure is being executed. For a detailed

discussion on how to use symbolic parameters in a set of control statements that you plan to catalog as a procedure, refer to the section "Using Symbolic Parameters."

page 117

Continuing Control Statements

When the total length of the fields on a control statement will exceed 71 columns, you must continue the fields onto one or more succeeding statements.

The command, comment, delimiter, and null statements cannot be continued.

You can continue the operand field or the comments field of other JCL statements. To continue either of these fields, you must follow the continuation conventions.

To continue the operand field:

1. Interrupt the field after a complete parameter or subparameter, including the comma that follows it, at or before column 71.
2. Comments can be included by following the interrupted field with at least one blank.
3. Code a nonblank character in column 72 when you are continuing a comments field. Optionally, code any nonblank character in column 72 for all other continued statements. If you do not code a character in column 72 when continuing the operand field, the system treats the next statement as a continuation statement as long as you follow the conventions outlined in items 4, 5, and 6.
4. Code the identifying characters // in columns 1 and 2 of the following statement.
5. Continue the interrupted operand beginning in any column from 4 through 16. If you begin coding after column 16, the system treats this statement as a comment field.
6. If there is a nonblank character (other than an asterisk) in column 3, the system assumes that this is a new statement. The job fails and an error message is issued saying no continuation is found.

To continue the comments field:

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

Any control statements in the input stream, other than a comment statement, that the system considers to contain only comments have //* in columns 1 through 3 on an output listing. Any control statements in a cataloged procedure, other than a comment statement, that the system considers to contain only comments have XX* in columns 1 through 3 on an output listing. For a comment statement, *** appears in columns 1 through 3 on an output listing. Any control statements in an instream procedure show ++ in columns 1 and 2 of an output listing.

Note: The examples in this publication have been formatted so that you can easily read them and they do not necessarily follow the rules for continuing control statements listed above.

Concatenating Data Sets

Up to 255 sequential or 16 partitioned input data sets, each of which may reside on a different volume, can be logically connected for the duration of a job step. To concatenate data sets, simply omit the ddnames from all the DD statements except the first in the sequence. When this ddname is encountered in a data control block in the processing program, each data set is automatically processed, in the same sequence as the DD statements defining them.

If concatenated data sets have unlike characteristics, for example, the device types, block lengths, or record format differ, the DCBOFLGS field of the data control block must be

modified while the program is executing. For details on concatenating data sets, refer to **OS/VS2 MVS Data Management Services Guide.**

If you make a backward reference to a concatenation (using an asterisk), the system obtains information only from the first data set defined in the sequence.

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

If you issue a read RDJFCB macro instruction to a DD statement that is concatenated, only the job file control block (JFCB) for the first data set is read.

You should not concatenate other data sets to a data set you have defined using the DUMMY parameter. When the processing program asks to read a dummy data set, an end-of-data-set exit is taken immediately and any concatenated data set is ignored.

Character Sets

Job control statements are coded using a combination of the characters from three different character sets. The contents of each of the character sets are described in Figure 18.

Character Set	Contents	
Alphameric	Alphabetic	A through Z
	Numeric	0 through 9
National	"At" sign	@
	Dollar sign	$
	Pound sign	#
Special	Comma	,
	Period	.
	Slash	/
	Apostrophe	'
	Left parenthesis	(
	Right parenthesis	)
	Asterisk	*
	Ampersand	&
	Plus sign	+
	Hyphen	-
	Equal sign	=
	Blank	

Figure 18. Character Sets

When coding any special characters, certain rules must be followed. These rules and the use of special characters are described next.

Using Special Characters

Special characters are used in the job control language to:

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

Sometimes you can code a special character that does not satisfy one of the three uses of special characters. In most of these cases, indicate that special characters are being used by enclosing the item that contains the special characters in apostrophes (5-8 punch), for

example, ACCT='123+456'. If one of the special characters is an apostrophe, you must code two consecutive apostrophes (two 5-8 punches) in its place, for example, 'O' 'NEILL'.

If one of the special characters is an apostrophe, you must code two consecutive apostrophes (two 5-8 punches) in its place, for example, 'O"NEILL'.

The following list contains those parameters that can have special characters as part of their variable information, and indicates when the apostrophes are not required.

- The accounting information on the JOB statement. The account number and additional accounting information can contain hyphens without being enclosed in apostrophes.
- The programmer's name on the JOB statement. The programmer's name can contain periods and/or hyphens without being enclosed in apostrophes. However, because a comma cannot immediately follow a period, the following is invalid: //JOBA JOB ,LEIGH.,TYPRUN=...
- The checkid field in the RESTART parameter on the JOB statement which may contain an asterisk.
- The ACCT parameter on the EXEC statement. The ACCT parameter can contain hyphens and plus zero (0-12 punch) without being enclosed in apostrophes.
- The PARM parameter on the EXEC statement may contain an ampersand for symbolic parameters. When the ampersand is used in that way, apostrophes are not required.
- The DSNAME parameter on the DD statement. The DSNAME parameter can contain hyphens without being enclosed in apostrophes. If the DSNAME parameter contains a qualified name, it can contain periods without being enclosed in apostrophes. If the DD statement identifies a generation of a generation data group, the generation number in the DSNAME parameter can contain a plus or minus (hyphen) sign without being enclosed in apostrophes. If the DD statement defines a temporary data set, the DSNAME parameter can contain, as the first two characters, ampersands without being enclosed in apostrophes. If the DD statement defines a member of a partitioned data set, a generation of a generation data group, or an area of an indexed sequential data set, the DSNAME parameter contains parentheses that enclose the member name, generation number, or area name; these parentheses are not enclosed in apostrophes.
- The volume serial number in the VOLUME parameter on the DD statement. The volume serial number can contain hyphens without being enclosed in apostrophes.
- Any subsystem-defined parameters that are coded with the SUBSYS parameter on the DD statement.
- The UNIT parameter on the DD statement. The device type on the UNIT parameter can contain hyphens (for example, 3330-1) without being enclosed in apostrophes.

When the parameters are not enclosed in apostrophes the following combinations are invalid:

- A period immediately after a left parenthesis or immediately before a right parenthesis, for example, DSN=(.ABC) or DSN=(ABC.)
- A period immediately followed by a comma.

Control Statement

The JOB statement marks the beginning of a job and, when jobs are stacked in the input stream, marks the end of the control statements for the preceding job.

```
//jobname   JOB     operands     comments
```

The JOB statement consists of the characters // in columns 1 and 2, and four fields — the name, operation (JOB), operands, and comments fields.

General Rules for Coding

- Code a JOB statement for each job. Code a unique jobname in every JOB statement. The jobname must consist of 1 through 8 alphameric and national (#,@,$) characters; the first character must be alphabetic or national.
- The two types of parameters that can be coded on the JOB statement are:

Positional parameters, which must precede any keyword parameters and must be coded in the following order.

accounting information
programmer's name

Keyword parameters, which can be coded in any order after the positional parameters. Any of the following keyword parameters can be coded on the JOB statement.

ADDRSPC
CLASS
COND
GROUP
MSGCLASS
MSGLEVEL
NOTIFY
PASSWORD
PERFORM
PRTY
RD
REGION
RESTART
TIME
TYPRUN
USER

- All parameters in the operand field are optional unless the account number and programmer's name parameters are required by the installation.
- If you code no parameters in the operand field of the JOB statement, code no comments.
- If you have installed MVS/System Product Release 2 (5740-XYN OR 5740-XYS) without installing Data Facility/Device Support Release 1 Enhancements (5740-AM7), do not use the checkpoint/restart facility.

If you have installed Data Facility/Device Support with either of the MVS/System Products you can use the checkpoint/restart facility with certain restrictions.

For additional information concerning these restrictions see **OS/VS2 MVS Checkpoint/Restart** and **OS/VS2 MVS System Programming Library: Job Management.**

Sample JOB Statements

```
//ALPHA     JOB     843,LINLEE,CLASS=F,MSGLEVEL=(1,1)
//LOS       JOB     ,BROWNLY,TIME=(4,30),MSGLEVEL=(2,0)
//MART      JOB     1863,RESTART=STEP4
//TRY8      JOB
//RACF1     JOB     'D83,123',USER=RAC01,GROUP=MYGROUP,PASSWORD=XYY
```

The Accounting Information Parameter—*positional, optional (according to installation procedures)*

The accounting information parameter identifies an account number and any information that may be required by the installation.

For information on how to add accounting facilities, refer to **OS/VS2 System Programming Library: System Management Facilities (SMF)**.

```
([account number][,additional accounting information,...])
```

account number
accounting number defined by your installation.
additional accounting information
additional information defined by your installation.

JES2 Parameters

JES2 assumes that the accounting information parameter contains information that could, alternatively, be specified on the /*JOBPARM statement. If the accounting information is specified in the indicated format, JES2 interprets and uses the subparameters of the accounting information parameter as follows:

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

pano
programmer's accounting number, one to four alphameric characters.
room
programmer's room number, one to four alphameric characters.
time
estimated execution time in minutes, up to four numeric digits (for example, ",30" for 30 minutes). If omitted, a standard value is assumed.
lines
estimated line count in thousands of lines, up to four numeric digits (for example, ",5" for 5000 lines). If omitted, a standard number of lines is assumed.
cards
estimated number of cards to be punched, up to four numeric digits. If omitted, a standard number of cards is assumed.
forms
special forms for printing an entire job, from one to four alphameric characters (for example, ",5" for five-part forms). If omitted, standard forms are assumed.
copies
number of times output is to be printed or punched, up to three numeric digits not exceeding an installation-specified limit (maximum 255) (for example, ",2" for two copies). If omitted, one copy is assumed.
log
Job Log option. This subfield should consist of one character. If this character is an "N", the JES2 Job Log is not produced. If any other character or if omitted, the log is produced unless NOLOG was specified for that job class during initialization.

linect
lines to be printed per page, up to three numeric digits not exceeding 255. If coded as zero, no automatic overflow is produced. If omitted, a standard value is assumed.

Note: JES2 may be initialized to ignore accounting field subparameters that are illegal by JES2 standards. If JES2 has been initialized to terminate a job that has an accounting field subparameter that is illegal by JES2 standards, then the first two subfields (**pano** and **room**) are required.

For a discussion of the JES2 scan of the accounting information parameter, refer to **System Programming Library: JES2 Installation, Initialization and Tuning.**

General Rules for Coding

- When accounting information consists of more than one item, enclose the information in either parentheses or apostrophes, for example, (5438,GROUP6) or '5438,GROUP6'. If you use apostrophes, all accounting information enclosed in the apostrophes is considered as one field.
- If you code an ampersand or an apostrophe as part of the accounting information, you must code them as double characters. For example, '2570"AB'.
- The account number and other accounting information must not exceed 142 characters in total, including the commas that separate the items.
- If you must continue the accounting information on another statement, enclose the information in parentheses.

Special Characters

- If any of the included items contain special characters (except hyphens), either:
  —Enclose the accounting information in apostrophes.
  —Enclose the item in apostrophes and the accounting information in parentheses. The enclosing apostrophes are not considered part of the information. If an apostrophe is part of the data, code two consecutive apostrophes.
- If one of the special characters is an ampersand or apostrophe and you are not defining a symbolic parameter, code two consecutive ampersands or apostrophes in its place.

Examples of the Accounting Information Parameter

```
//JOB43     JOB     D548-8686
```

```
//JOB44     JOB     (D548-8686,'12/8/73',ERICKSON)
```

Accounting number plus additional information; parentheses are required.

```
//JOB45     JOB     (CFH1,2G14,15,2)
```

With JES2: programmer's accounting number, room number, estimated job time, plus estimated line count; parentheses are required. Standard values are assumed for remaining JES2 parameters.

The ADDRSPC Parameter—*keyword, optional*

The ADDRSPC parameter indicates whether or not the job can be paged.

For further information on the ADDRSPC parameter, see the section "Requesting Storage For Execution of a Program."

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

VIRT
a keyword indicating that the job can be paged.

REAL
a keyword indicating that the job cannot be paged.

Default: If you omit the ADDRSPC parameter, the default is VIRT.

General Rules for Coding

- The ADDRSPC parameter coded on a JOB statement will override any ADDRSPC parameter coded on an EXEC statement for that job.
- Requests for real and virtual storage can be made in the same job. Each step will honor the request for either real or virtual storage if there is no ADDRSPC parameter on the JOB statement.

Rule for Coding when Using Real Storage

Code the REGION parameter to specify how much real storage is needed, or allow it to default.

Rule for Coding when Using Virtual Storage

The installation provides a default region size for use when the REGION parameter is not specified when ADDRSPC=VIRT is coded or implied. That value, or the value specified on the REGION parameter if coded, sets the upper boundary to limit region size for variable-length GETMAINs as long as there remains in the region at least the minimum amount of storage requested. In addition, the region value is used by an IBM or installation-supplied routine (IEALIMIT) to establish a second value used to limit fixed-length GETMAINs, and variable-length GETMAINs when the space remaining in the region is less than the requested minimum. When the minimum requested length for variable-length GETMAINs causes this second value to be exceeded, the job or job step abnormally terminates. For further information, see **OS/VS2 System Programming Library: Supervisor.**

Examples of the ADDRSPC Parameter

```
//PEH       JOB      ,BAKER,ADDRSPC=VIRT
```

The ADDRSPC parameter requests virtual storage. The area size available to the user is the installation-supplied default, since a value is not specified on the REGION parameter.

```
//DEB       JOB      ERIC,ADDRSPC=REAL,REGION=100K
```

The ADDRSPC parameter requests real storage. The REGION parameter specifies the amount; in this case, 100K.

The CLASS Parameter—*keyword, optional*

The CLASS parameter assigns a job class to each job, depending on the characteristics of the job and the installation's rules for assigning a job class.

For further information on the use of the CLASS parameter, see "Assigning a Job to a Job Class."

```
CLASS=jobclass
```

jobclass
any character A-Z or 0-9, defined by the installation.

Default: Determined by the source of the job; that is, the particular card reader or work station, or whether the job was submitted by a time-sharing user. JES2 and JES3 use the installation-defined default.

Rule for Coding

When coding JES3, a valid CLASS parameter on the JES3 MAIN statement overrides a valid CLASS parameter on the JOB statement.

Example of the CLASS Parameter

```
//SETUP     JOB     1249,CORNER,CLASS=M
```

This job is assigned to class M.

The COND Parameter—*keyword, optional*

The COND parameter specifies whether a job will continue processing based on return codes issued by one or more of its job steps. Each test specified by the COND parameter is performed using the return code of a completed job step. If any of the tests are satisfied, the remaining job steps are bypassed and the job is terminated.

For further information on the use of the COND parameter, see "Conditional Execution of Job Steps."

```
COND=((code,operator),...)
```

code
a decimal number from 0 through 4095. This number is compared with the return code issued by each job step.

Note: Specifying a decimal number greater than 4095 could result in invalid return code testing or invalid return codes in messages.

operator
the type of comparison to be made with the return code. Operators and their meanings are:

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

Rules for Coding

- If you code only one return code test, you need not code the outer parentheses.
- You can code up to eight different return code tests for each job. If specifying more than eight tests, a JCL error message is issued and the job abnormally terminates.
- If you code the COND parameter on the JOB statement and on one or more of the job's EXEC statements, and the return code test requested on the JOB statement is satisfied, the job terminates even if the return code test requested on the EXEC statement is not satisfied. Likewise, if the return code test requested on the EXEC statement is satisfied, the step is bypassed even if the test requested on the JOB statement is not satisfied.

Examples of the COND Parameter

```
//TYPE      JOB     (611,402),BOURNE,COND=(7,LT)
```

If 7 is less than the return code, the job is terminated. (Any return code less than or equal to 7 allows the job to continue.)

```
//TEST      JOB     501,BAXTER,COND=((20,GE),(30,LT))
```

If 20 is greater than or equal to the return code, or 30 is less than the return code, the job is terminated. (Any code of 21 through 30 allows the job to continue.)

The GROUP Parameter — *keyword, optional*

The GROUP parameter is used to allow a RACF-defined user or users to share access to RACF-protected resources.

```
GROUP=group name
```

group name
1 to 8 alphameric or national characters which identify the group that the user is associated with during the job. The first character must be alphabetic or national.

Default: If this parameter is omitted and the USER and PASSWORD parameters are coded, the default group for the user is used.

Rule for Coding

If GROUP is coded, the USER and PASSWORD parameters must also be specified.

Example of the GROUP Parameter

```
//TEST      JOB     'D83,123456',GROUP=MYGROUP,USER=MYNAME,PASSWORD=ABC
```

Requests that the RACF-defined user be associated with the group named MYGROUP for the duration of the job.

The MSGCLASS Parameter—*keyword, optional*

The MSGCLASS parameter specifies the output class to which system messages and JCL statements for the job are to be written.

For further information on use of the MSGCLASS parameter, see "Obtaining Output" for either JES2 or JES3.

```
MSGCLASS=output class
```

output class
an alphabetic (A-Z) or numeric (0-9) character.

Default: Determined by the source of the job; that is, the particular card reader or work station, or whether the job was submitted by a time-sharing user.

Rule for Coding

System messages and output data sets can be routed to the same output class. To do this, code the same output class in both the MSGCLASS parameter on the JOB statement and the SYSOUT parameter on the DD statements for the data sets. Or, code SYSOUT=* on all DD statements for the SYSOUT data sets you want to default to the MSGCLASS output class of the job.

Examples of the MSGCLASS Parameter

```
//IN          JOB     GEORGE,MSGCLASS=F
```

Specifies an output class.

```
//BOTTLE          JOB     MENTLE,MSGLEVEL=(2,0)
```

Specifies no output class. In this case, the output class defaults to the MSGCLASS value specified by your installation.

```
//A1403     JOB     MSGCLASS=L
//STEP1     EXEC    PGM=PRINT
//OUTPUT    DD      SYSOUT=L
```

Specifies that a job's system messages MSGCLASS parameter and output data set SYSOUT parameter are to be routed to the same output class.

The MSGLEVEL Parameter—*keyword, optional*

The MSGLEVEL parameter indicates what job output is to be written as part of the output listing. The following output can be requested:

- The JOB statement
- All input job control statements
- All cataloged procedure statements for procedures called by any of the job's steps and the internal representation of procedure statement parameters after symbolic parameter substitution.
- Allocation, disposition, and allocation recovery messages (allocation/termination messages.)

For further information on the MSGLEVEL parameter, see "Obtaining Output" for either JES2 or JES3.

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

statements
a number that indicates which job control statements are to be written as output from a job. The choices are:

0 only the JOB statement is to be written.
1 all input job control statements, cataloged procedure statements, and the internal representation of procedure statement parameters after symbolic parameter substitution are to be written.
2 only input job control statements are to be written.

messages
a number that indicates what allocation/termination messages are to be written. Choices are:

0 no allocation/termination messages are to be written, unless the job terminates abnormally.
1 allocation/termination messages are to be written.

Default: For JES2, the default is associated by job class. For JES3, the default is associated by the operator with the particular reader. For further information on specifying default values, refer to **System Programming Library: JES2 Installation, Initialization and Tuning** or **JES3 System Programming Library: Installation Planning and Tuning.**

Rule for Coding

If the second (messages) subparameter is omitted, you do not need to code the parentheses.

Examples of the MSGLEVEL Parameter

```
//GD40      JOB     GEORGE,MSGLEVEL=(2,1)
```

Requests that only input statements and all allocation/termination messages be written.

```
//PAUL      JOB     MENTLE,MSGLEVEL=0
```

Requests that only the JOB statement be written.

The NOTIFY Parameter—*keyword, optional*

The NOTIFY parameter is used to request that a message be sent to a user's time-sharing terminal when his background job has completed processing.

```
NOTIFY=user identification
```

user identification
a 1 to 7 alphameric identification of the user to be notified. The first character must be alphabetic.

Rules for Coding

- Code the same user identification that you specify when you start a terminal session (at LOGON).
- **Receiving notification that the job has completed processing.** For JES2, if you are logged on to any processor, you will be immediately notified when the job completes. If you are not logged on, the message will be saved and presented when you log on to the system you originally submitted the job from. For JES3, if you are logged on, you will be notified when the job completes. If you are not logged on, the message will be saved and presented when you log on to the system you originally submitted the job from. To receive notification of the completion of a job submitted through batch processing, supply a main-name on the ACMAIN parameter of the MAIN statement in addition to specifying NOTIFY on the JOB statement. The ACMAIN parameter identifies the main processor that your TSO system is running on.

Example of the NOTIFY Parameter

```
//SIGN      JOB     NOTIFY=POK1,MSGLEVEL=(2,1)
```

When the job "SIGN" has completed processing, a message will be sent to the user "POK1" informing him that his job has completed processing.

The PASSWORD Parameter — *keyword, optional*

The PASSWORD parameter is used to identify a current RACF password or specify a new RACF password.

```
PASSWORD=(password[,new password])
```

password
1 to 8 alphameric or national characters identifying the user's current password.

new password
1 to 8 alphameric or national characters identifying the user's new password.

Rules for Coding

- If PASSWORD is coded, the USER parameter must also be specified.
- PASSWORD is required if USER or GROUP is specified.
- Parentheses are not needed if the new password subparameter is omitted.
- You can specify a new password at any time. You **must** specify a new password when the current password has expired.

Examples of the PASSWORD Parameter

```
//TEST1     JOB     'D83,123456',PASSWORD=ABCDE,USER=MYNAME
```

Identifies ABCDE as the current password for the RACF user.

```
//TEST2     JOB 'D83,123456',PASSWORD=(BCH,A12@),USER=RAC1,GROUP=GRP1
```

Requests that the current password BCH be changed to A12[.

The PERFORM Parameter—*keyword, optional*

The PERFORM parameter specifies the performance group definition with which a job is associated.

For information on the performance groups, see "Performance of Jobs and Job Steps" for either JES2 or JES3.

```
PERFORM=n
```

n
a number between 1 and 999 inclusive, identifying a performance group that has been defined by the installation.

Default: The interpreter obtains a default from the system resources manager and issues a warning message indicating that a system default is set.

- For non-TSO jobs, the IBM-supplied default is one (1).
- For TSO sessions, the IBM-supplied default is two (2).
- An installation-supplied or IBM-provided default can override the performance group number you code if it fails to pass validity checks. The **OS/VS2 SPL: Initialization and Tuning Guide** contains a detailed description of the override procedure.

Note: Consult your systems programmer to find out which performance groups are valid and which performance groups apply to which job types.

Rules for Coding

- If PERFORM is specified on the JOB statement, its value supersedes any PERFORM specifications on EXEC statements associated with the job.
- If no PERFORM parameter appears on the JOB statement and a PERFORM parameter appears on an associated EXEC statement, the parameter value appearing on the EXEC statement is used during the associated job step.

Example of the PERFORM Parameter

```
//STEP1     JOB     MARLA,CLASS=D,PERFORM=25
```

Class D determines in which class the job will be executed. Once in the system, the job will run in performance group 25 if it passes validity checks. The significance of this performance group is defined by the installation.

Example of the PERFORM Parameter

```
//STEP1     JOB     MARLA,CLASS=D,PERFORM=25
```

The following information applies to MVS/System Extensions Release 2.

Class D determines in which class the job will be executed. Once in the system, the job will run in performance group 25 if it passes validity checks. The significance of this performance group is defined by the installation.

The following information applies to only MVS/System Extensions Release 1.

Class D determines in which class the job will be executed. Once in the system, the job will run in performance group 25. The significance of this performance group is defined by the installation.

Programmer's Name Parameter—*positional, optional (according to installation procedures)*

The programmer's name parameter identifies the person or group responsible for a job.

```
programmer's name
```

Rules for Coding

- If the programmer's name parameter is coded, place it after the accounting information parameter.
- Code the programmer's name parameter before any or all keyword parameters.
- The programmer's name must not exceed 20 characters, including all special characters.
- When the programmer's name contains special characters, other than hyphens, or leading or imbedded periods, enclose the name in apostrophes. If the last character of the name is a period, the name must also be enclosed in apostrophes. If the special characters include apostrophes, each apostrophe must be coded as two consecutive apostrophes.
- If the programmer's name parameter is not required, you need not code a comma to indicate its absence.

Examples of the Programmer's Name Parameter

```
//APP       JOB     ,C.L.BROWN
```

Programmer's name, no accounting information. (The leading comma may be optional. Check with your installation.)

```
//DELTA         JOB     'T.O''NEILL'
```

Programmer's name containing special characters, no accounting information required. (An apostrophe as part of the name must be coded as two consecutive apostrophes and the entire name then enclosed in single apostrophes.)

```
//#308      JOB     (846349,GROUP12),MATTHEW
```

Account number plus additional accounting information and programmer's name.

```
//JOBA          JOB     'NICOLLE.',TYPRUN=HOLD
```

Apostrophes must be used if the last character in the name is a period.

The PRTY Parameter—*keyword, optional* (JES2)

The PRTY parameter specifies a job's queue selection priority: the priority by which the job is selected from all internal JES2 queues except hardcopy. For further information about this priority, see "Routing a Job Through the System JES2)".

Note: Depending on the JES2 initialization options specified, the PRTY parameter might be ignored. For further information on PRTY see **System Programming Library: JES2 Installation, Initialization, and Tuning.**

```
PRTY=priority
```

priority
a number from 0 to 15 indicating the job's queue selection priority. The highest priority is 15.

Default: Installation default.

Rule for Coding

If the priority value is not between 0 and 15, it is ignored by JES2.

Example of the PRTY Parameter

```
//JOBA      JOB     1,'EXAMPLE JOB',PRTY=12
```

The job has a queue selection priority of 12.

The PRTY Parameter—*keyword, optional* (JES3)

The PRTY parameter specifies a job's initiation or selection priority within its job class group.

For further information, see "Routing a Job Through the System (JES3)".

```
PRTY=priority
```

priority
a number from 0 to 13 indicating a job's priority. The highest priority is 13.

Default: Installation default.

Rule for Coding

If the priority value is not between 0 and 13, an error will occur.

Examples of the PRTY Parameter

```
//#1930     JOB     RICHARDSON,CLASS=C,PRTY=8
```

The job will have an initiation priority of 8 in the job class C.

```
//RING      JOB     WILLIAMS,PRTY=4
```

The job will have an initiation priority of 4.

The RD Parameter—*keyword, optional*

The RD (restart definition) parameter specifies how the step restart facilities are used with the CHKPT macro instruction, and whether automatic restart is permitted or suppressed.

For detailed information on the checkpoint/restart facility, refer to **OS/VS2 MVS Checkpoint/Restart.**

RD={R / RNC / NC / NR}

R
indicates that automatic step restart is permitted.
If the processing program used by a job step does not include a CHKPT macro instruction, coding RD=R allows execution to be resumed at the beginning of the abnormally terminated step.

If the program does include a CHKPT macro instruction, coding RD=R permits automatic step restart if the step abnormally terminates before execution of the CHKPT macro instruction; thereafter, only checkpoint restart can occur.

If you cancel the effects of the CHKPT macro instruction before a checkpoint restart is performed, the request for automatic step restart is again in effect.

RNC
indicates that automatic step restart is permitted and automatic and deferred checkpoint restart are not permitted.

NC
indicates that automatic step restart and automatic and deferred checkpoint restart are not permitted.

NR
indicates that a CHKPT macro instruction can establish a checkpoint, but neither automatic step restart nor automatic checkpoint restart is permitted if the job fails. (If you have coded RD=NR and a system failure occurs, RD=NR does not prevent the job from restarting.) Coding RD=NR allows you to resubmit the job at a later time and specify in the RESTART parameter (on the JOB statement of the resubmitted job) the checkpoint at which execution is to be resumed.

Rules for Coding

- Automatic restart will not be honored if you do not have a job journal (a job journal is an initialization option).
- If you specify RD=R or RD=RNC on either the JOB or EXEC statements, JES2 forces journaling.
- Even if you do not code the RD parameter, the job is eligible for automatic checkpoint/restart if checkpoints have been issued. However, the job is not eligible for automatic step restart.
- If you code the RD parameter on the JOB statement, any RD parameters coded on the job's EXEC statements are ignored and the value coded on the JOB statement is effective for all steps.
- The RD parameter values NC and RNC can be used to suppress the action of the CHKPT parameter.

Examples of the RD Parameter

```
//JILL      JOB     333,ERIC,CLASS=C,RD=R,MSGLEVEL=(1,1)
```

Permits execution to be automatically restarted with a step that abnormally terminates.

```
//TRY56     JOB     333,ERNEST,RD=RNC,MSGLEVEL=(1,1)
```

Permits execution to be automatically restarted beginning with a step that abnormally terminates; suppresses the action of CHKPT macro instruction.

```
//PASS      JOB     (721,994),PEOPLE,RD=NR,MSGLEVEL=(0,1)
```

Neither automatic step nor checkpoint restart can occur, but CHKPT macro instruction can establish checkpoints.

The REGION Parameter—*keyword, optional*

The REGION parameter specifies the amount of space to be allocated to a job.

For further information on the REGION parameter, see the section "Requesting Storage For Execution of a Program."

```
REGION=valueK
```

valueK
a number that indicates how many bytes of storage are to be allocated to a job.

Default: the installation-defined default; by job class in JES2 and operator with the particular reader in JES3.

General Rules for Coding

- Code an even number. (If you code an odd number, the system treats it as the next highest even number.)
- When you want to specify a different region size for each job step, code the REGION parameter on the EXEC statements, instead of the JOB statement.
- If you code the REGION parameter on the JOB statement, REGION parameters coded on the job's EXEC statements are ignored.
- If you code the region value as 0, or allow it to default to 0, results are unpredictable.

Rule for Coding when Using Virtual Storage

The installation provides a default region size for use when the REGION parameter is not specified when ADDRSPC=VIRT is coded or implied. That value, or the value specified on the REGION parameter if coded, sets the upper boundary to limit region size for variable-length GETMAINs as long as there remains in the region at least the minimum amount of storage requested. In addition, the region value is used by an IBM or installation-supplied routine (IEALIMIT) to establish a second value used to limit fixed-length GETMAINs, and variable-length GETMAINs when the space remaining in the region is less than the requested minimum. When the minimum requested length for variable-length GETMAINs causes this second value to be exceeded, the job or job step abnormally terminates. For further information, see **OS/VS2 System Programming Library: Supervisor.**

Examples of the REGION Parameter

```
//SANDY     JOB     A23,COFFEE,REGION=100K,ADDRSPC=REAL
```

100K of real storage is assigned.

```
//ACCT4     JOB     175,FRED,REGION=250K
```

A 250K region is assigned. For variable-length GETMAINs, the request is satisfied as long as there remains in the region at least the minimum amount requested. If less than the minimum remains, the minimum is allocated as long as the allocation does not cause the second limiting value (described above) to be exceeded. For fixed-length GETMAINs, the amount requested is allocated as long as the allocation does not cause the second limiting value to be exceeded.

The RESTART Parameter—*keyword, optional*

The RESTART parameter indicates that the restart facilities are to be used to resubmit a job for execution. Execution can be restarted at the beginning of a step (step restart) or within a step (checkpoint restart).

For detailed information on the checkpoint/restart facilities, refer to **OS/VS2 MVS Checkpoint/Restart.**

```
RESTART=( {*                    } [,checkid] )
          {stepname             }
          {stepname.procstepname}
```

*****
indicates that execution is to be restarted at or within the first job step.

stepname
specifies that execution is to be restarted at or within the named job step.

stepname.procstepname
specifies that execution is to be restarted at or within a cataloged procedure step. Stepname is the name of the job step that calls the cataloged procedure, and procstepname is the name of the procedure step.

checkid
specifies the name of the checkpoint at which execution is to be restarted. When checkid is coded, execution is restarted within the specified job step at the named checkpoint. If checkid is not coded, execution is restarted at the specified job step.

Rules for Coding

- Code * in place of stepname.procstepname if the first job step calls a cataloged procedure and execution is to be restarted at or within the first procedure step.
- The first character of stepname must be alphabetic.
- You need not code the parentheses if execution is to be restarted at a job step; that is, if you do not code the checkid subparameter.
- If the checkpoint name contains special characters, the name must be enclosed in apostrophes. If one of the special characters is an apostrophe, identify it by coding two consecutive apostrophes in its place.
- Include the SYSCHK DD statement when execution is to be restarted within a job step. (The SYSCHK DD statement is described in the section on DD statements in this publication.)
- Before resubmitting a job, check all backward references to steps that precede the restart step. Eliminate all backward references for the following keywords: PGM on the EXEC statements and VOLUME=REF=reference on the DD statements. A backward reference of VOLUME=REF=reference is allowed if the statement to which reference is made includes VOLUME=SER=(serial number,...).
- Before resubmitting a job, all EXEC statements that contain the COND parameter should be carefully reviewed. If any of the COND parameters contain values referencing a step preceding the restart step, be aware that the COND parameters will be ignored for these steps.

Generation Data Sets

In the restart step or in steps following it, do not use the original relative generation numbers to refer to generation data sets that were created and cataloged in steps preceding the restart step. Instead, refer to a generation data set by its present relative generation number. For

example, if the last generation data set created and cataloged was assigned a generation number of +2, refer to it as 0 in the restart step and in steps following the restart step. In this case, refer to the generation data set assigned a generation number of +1 as -1.

If generation data sets created in the restart step were kept instead of cataloged (that is, DISP=(NEW,CATLG,KEEP) was coded and abnormal termination occurred), refer to generation data sets during checkpoint restart by the same relative generation numbers that were used to create them.

Examples of the RESTART Parameter

```
//LINES     JOB     RESTART=COUNT
```

Specifies that execution is to be restarted at the job step name COUNT.

```
//@LOC5     JOB     RESTART=(PROCESS,CHKPT3)
//SYSCHK    DD      DSN=CHK,UNIT=3330,DISP=OLD
```

Specifies that execution is to be restarted within the job step named PROCESS at the checkpoint named CHKPT3. This JOB statement must be followed by a DD statement named SYSCHK, which defines the data set on which an entry for the checkpoint name CHKPT3 was written.

```
//WORK      JOB     RESTART=(*,CKPT2)
//SYSCHK    DD      DSNAME=CHKPT,UNIT=3330,DISP=OLD
```

Specifies that execution is to be restarted at the checkpoint name CKPT2 in the first job step. This JOB statement is followed by a DD statement name SYSCHK, which defines the data set on which an entry for the checkpoint name CKPT2 was written.

```
//CLIP5     JOB     RESTART=(PAY.WEEKLY,CHECK8)
//SYSCHK    DD      DSN=CHKPT,UNIT=2314,DISP=OLD
```

Specifies that execution is to be restarted within the procedure step named WEEKLY at the checkpoint name CHECK8. PAY is the name of the job step that calls the cataloged procedure that contains the procedure step named WEEKLY. This JOB statement must be followed by a DD statement named SYSCHK, which defines the data set on which an entry for the checkpoint named CHECK8 was written.

The TIME Parameter—*keyword, optional*

The TIME parameter specifies the maximum amount of time that a job may use the CPU. The CPU time used by the job is written on the output listing.

```
TIME=      {([minutes][,seconds])}
           {1440                 }
```

minutes
a number that specifies the maximum number of minutes the job can use the CPU. The number of minutes must be less than 1440 (24 hours).

seconds
a number that specifies the maximum number of seconds beyond the specified number of minutes the job can use the CPU, or if no minutes are specified, the maximum number of seconds the job can use the CPU. The number of seconds must be less than 60.

1440
specifies that the job is not to be timed.

Default: When you do not specify TIME on the JOB statement or the JES2 /*JOBPARM statement JES2 uses the default value specified at JES2 initialization.

Rules for Coding

- If you code the CPU time limit in minutes only, you need not code the parentheses.
- Code 1440 if the job can use the CPU for 24 hours or more or if any of the job's steps should be allowed to remain in a wait state for more than the established time limit.
- Because the CPU time-used field is checked at intervals of about 10.5 seconds, the actual amount of time that a job uses the CPU can exceed the time specified on the TIME parameter by up to 10.5 seconds.
- A job that exceeds the specified limit causes termination of the job unless you use a user exit routine to extend the time.
- TIME=0 is not supported. Results are unpredictable when used on the JOB statement.
- If the TIME parameter is coded on the JOB statement with a value other than 1440, the time limit for each step is set to the step time limit (the value coded on the TIME parameter of the EXEC statement or the limit specified by the installation) or the remaining job time, whichever is smaller.
- If the TIME parameter is not coded on the JOB statement, each job step is timed individually according to the value coded on the TIME parameter of the EXEC statement or the limit specified by the installation.
- The time limit specified for a job or the time remaining for successive steps in a multistep job is converted, by the system, to seconds and then rounded to the nearest unit (1 unit=1.048576 seconds). Thus a step can begin execution with up to one-half unit more or one-half unit less time than expected. For example, if the time remaining for the job is less than one-half unit, a step will begin execution with zero time, resulting in an abnormal termination.

Examples of the TIME Parameter

```
//SEED      JOB     TIME=(12,10)
```

Specifies that the maximum amount of time the job can use the CPU is 12 minutes, 10 seconds.

```
//TYPE41    JOB     TIME=(,30)
```

Specifies that the maximum amount of time the job can use the CPU is 30 seconds.

```
//FORMS     JOB     TIME=5
```

Specifies that the maximum amount of time the job can use the CPU is 5 minutes.

```
//RAINCK    JOB     TIME=1440
```

Specifies that the job is not timed. Therefore, the job may use the CPU and may remain in wait state for an unspecified period of time.

Specifies that the job can have unlimited use of the CPU. Therefore, the job can use the CPU and can remain in wait state for an unspecified period of time.

Examples of Coding the Time Parameter on the JOB and EXEC Statements

---

```
//FIRST JOB    TIME=2
//STEP1 EXEC       TIME=1
//STEP2 EXEC       TIME=1
```

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

```
//SECOND    JOB     TIME=3
//STEP1 EXEC        TIME=2
//STEP2 EXEC        TIME=2
```

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

The TYPRUN Parameter—*keyword, optional*

The TYPRUN parameter specifies special JES2 and JES3 processing. For TYPRUN=HOLD or TYPRUN=JCLHOLD, the operator must be informed of what event is to occur. When the event has occurred, the operator releases the job from hold status, thereby allowing it to be selected for processing. The TYPRUN parameter can also specify that the JCL for a job be scanned for syntax errors and, for JES2, that the input deck is converted directly to a SYSOUT data set and scheduled for output processing.

For further information on the TYPRUN parameter, see "Delaying Job Initiation" and "Bypassing Job Initiation."

```
          ⎧HOLD   ⎫
          ⎪JCLHOLD⎪
TYPRUN=   ⎨SCAN   ⎬
          ⎩COPY   ⎭
```

HOLD
specifies that the job is to be held prior to execution until the operator releases the job. However, if an error occurs during input service processing, the job will not be held, but will be scheduled for Reader-Interpreter/Converter-Interpreter processing (JES3) or output and purge processing (JES2).

JCLHOLD
(for JES2) specifies that the job is to be held before processing by the JCL converter until the operator releases the job.

SCAN
specifies that the JCL for a job is to be scanned for syntax errors but that the job is not to be executed. For example, SCAN checks for invalid keywords, illegal characters, and parentheses errors. SCAN does not check for parameter value errors or excessive parameters in JES2 but does check for these in JES3. SCAN does not check for misplaced statements and does not check the syntax of JCL statements included in cataloged procedures referenced by the job.

COPY
(for JES2) specifies that the input deck (as submitted) is converted directly to a SYSOUT data set and scheduled for output processing. The class of the SYSOUT data set is the same as the message class of the job and can be controlled by the MSGCLASS parameter. This feature is available in JES3 by using nonstandard job processing. (See the section, "The PROCESS Statement" for information concerning nonstandard job processing.)

Examples of the TYPRUN Parameter

```
//UPDATE    JOB
//LIST      JOB     TYPRUN=HOLD
```

Jobs UPDATE and LIST are to be submitted for execution. The job UPDATE uses a program that adds and deletes members to a library; the job LIST uses a program that lists the members of a library. For an up-to-date listing of the library, UPDATE must be executed before LIST. This is accomplished by coding TYPRUN=HOLD on the JOB statement for the job named LIST. If a MONITOR JOBNAMES command is issued by you or the operator, the operator is notified on the console when UPDATE has completed processing. The operator releases LIST, which can then be selected for execution.

```
//UPDTPROC  JOB
//TESTPROC  JOB     TYPRUN=JCLHOLD
```

Jobs UPDTPROC and TESTPROC are to be submitted for execution. The job UPDTPROC adds a cataloged procedure to SYS1.PROCLIB; the JCL in job TESTPROC invokes the new procedure. The second JOB statement specifies TYPRUN=JCLHOLD so that TESTPROC will not be processed by the JCL converter until the required procedure has been added to the library by UPDTPROC. The operator releases TESTPROC after UPDTPROC has executed.

The USER Parameter—*keyword, optional*

The USER parameter is used to specify a RACF userid for identification.

```
USER=userid
```

userid
1 to 7 alphameric or national characters that identify the RACF-defined user to the system. The first character must be alphabetic or national.

Default: If this parameter and the GROUP parameter are omitted, RACF assigns a default userid and group name.

Rules for Coding

- If USER is coded, the PASSWORD parameter must also be specified.
- USER is required if GROUP or PASSWORD is specified.

Example of the USER Parameter

```
//TEST      JOB     'D83,123456',USER=MYNAME,PASSWORD=ABCD
```

Identifies MYNAME as the userid for this RACF-defined user.

Control Statement

The EXEC statement is the first statement of each job step and cataloged procedure step. It identifies the program to be executed or the cataloged procedure to be called.

```
//stepname  EXEC    operands     comments
```

The EXEC statement consists of the characters // in columns 1 and 2 and four fields — the name, operation (EXEC), operand, and comments fields.

General Rules for Coding

- Code an EXEC statement for each job step.
- A stepname is optional. However, when you want to perform certain functions, you should code a valid and unique stepname in the name field for each job step within the job. A stepname is necessary to:
  - Make backward references to the step.
  - Override parameters on an EXEC statement or DD statement in a cataloged procedure step, and add DD statements to a cataloged procedure step.
  - Perform a step or checkpoint restart at or within the step.
- The stepname must consist of 1 through 8 alphameric and national (@,#,$) characters. The first character must be an alphabetic or national character.
- When [.procstepname] immediately follows a keyword parameter, the parameter applies only to the specified step within the procedure invoked by the EXEC statement.
- The two types of parameters that can be coded in the operand field of the EXEC statement are:

Positional parameters, which must precede any keyword parameter. One of the following three positional parameters must be coded:

PGM
PROC
procedure name

Keyword parameters, which can be coded in any order after the positional parameter. Any of the following keyword parameters can be coded on the EXEC statement:

ACCT
ADDRSPC
COND
DPRTY
DYNAMNBR
PARM
PERFORM
RD
REGION
TIME

If you have installed MVS/System Product Release 2 (5740-XYN or 5740-XYS) without installing Data Facility/Device Support Release 1 Enhancements (5740-AM7) do not use the checkpoint/restart facility.

If you have installed Data Facility/Device Support with either of the MVS/System Products you can use the checkpoint/restart facility with certain restrictions.

For additional information concerning these restrictions see **OS/VS2 MVS Checkpoint/Restart** and **OS/VS2 MVS System Programming Library: Job Management.**

Sample EXEC Statements

```
//STEP4     EXEC    PGM=DREC,PARM='3018,NO'
//          EXEC    PGM=ENTRY,TIME=(2,30)
//FOR       EXEC    PROC=PE489,TIME=4
```

The ACCT Parameter—*keyword, optional*

The ACCT parameter specifies one or more subparameters of accounting information to be passed to the installation's accounting routines.

For further information concerning the accounting routines, see **OS/VS2 MVS System Programming Library: System Management Facilities (SMF).**

```
ACCT[.procstepname]=(accounting information,...)
```

accounting information
specifies one or more subparameters established by the installation as accounting information to be passed to the accounting routines.

Rules for Coding

- If the accounting information consists of only one subparameter, you need not code the parentheses.
- The maximum number of characters of accounting information, including the commas that separate the subparameters, is 142.
- If a subparameter contains special characters (other than hyphens), enclose the subparameter in apostrophes. The apostrophes are not considered part of the information. If one of the special characters is an apostrophe, code two consecutive apostrophes in its place.
- If the job step calls a cataloged procedure, the ACCT parameter overrides any ACCT parameters coded in the procedure steps. This pertains to all procedure steps.
- If different steps in a cataloged procedure required different accounting information, code ACCT.procstepname=(accounting information,...) for each step that requires unique accounting information. Accounting information will then pertain only to the named procedure step.

Examples of the ACCT Parameter

```
//STEP1     EXEC    PGM=JP5,ACCT=(LOCATION8,'CHGE+3')
```

Specifies that this accounting information pertains to this job step.

```
//STP3      EXEC    LOOKUP,ACCT=('/83468')
```

Specifies that this information pertains to this job step. Since this step calls a cataloged procedure, the accounting information pertains to all the steps in the procedure.

```
//STP4      EXEC    BILLING,ACCT.PAID=56370,ACCT.LATE=56470,
//                  ACCT.BILL='121+366'
```

Specifies that different accounting information pertains to each of the named procedure steps (PAID,LATE, and BILL).

The ADDRSPC Parameter—*keyword, optional*

The ADDRSPC parameter indicates whether the job step will use virtual or real storage, that is, whether or not the job step can be paged.

For further information on the ADDRSPC parameter, see the section "Requesting Storage For Execution of a Program".

```
ADDRSPC[.procstepname]={VIRT }
                       {REAL }
```

VIRT
a keyword indicating that the job step can be paged to virtual storage.

REAL
a keyword indicating that the job step cannot be paged. The job step must exist in real storage.

Default: If you omit the ADDRSPC parameter, the default is VIRT.

General Rules for Coding

- The ADDRSPC parameter coded on a JOB statement will override any ADDRSPC parameter coded on an EXEC statement for that job.
- Requests for real and virtual storage can be made in the same job. Each step will honor the request for either real or virtual storage if there is no ADDRSPC parameter on the JOB statement.

Rule for Coding when Using Real Storage

Code the REGION parameter to specify how much storage is needed, or allow it to default.

Rule for Coding when Using Virtual Storage

The installation provides a default region size for use when the REGION parameter is not specified when ADDRSPC=VIRT is coded or implied. That value, or the value specified on the REGION parameter if coded, sets the upper boundary to limit region size for variable-length GETMAINs as long as there remains in the region at least the minimum amount of storage requested. In addition, the region value is used by an IBM or installation-supplied routine (IEALIMIT) to establish a second value used to limit fixed-length GETMAINs, and variable-length GETMAINs when the space remaining in the region is less than the requested minimum. When the minimum requested length for variable-length GETMAINs causes this second value to be exceeded, the job or job step abnormally terminates. For further information, see **OS/VS2 System Programming Library: Supervisor.**

Examples of the ADDRSPC Parameter

```
//CAC1      EXEC    A,ADDRSPC=VIRT
```

The ADDRSPC parameter requests virtual storage. The area size available to the user is the installation-supplied default, or the region size specified on the JOB statement, since the REGION parameter is not coded.

```
//CAC2      EXEC    B,ADDRSPC=REAL,REGION=80K
```

The ADDRSPC parameter requests real storage. The REGION parameter specifies the amount, in this case, 80K.

The COND Parameter—*keyword, optional*

also see p. 134

The COND parameter specifies whether a job step will be executed based on return codes issued by one or more of the preceding job steps. This parameter allows the specification of conditions for bypassing a job step, as well as for executing a job step.

Each test specified by the COND parameter is performed using the return code of a completed job step. If any of the tests are satisfied, the particular job step is bypassed.

For further information on the use of the COND parameter, see "Conditional Execution of Job Steps." (p. 41 or 73)

```
COND[.procstepname]= ( {(code,operator)                          } ,...  [,EVEN  ])
                       {(code,operator,stepname)                 }       [,ONLY  ]
                       {(code,operator,stepname.procstepname)    }
```

code
a number from 0 through 4095. This number is compared with the return code issued by all previous steps or a specific step.

Note: Specifying a decimal number greater than 4095 could result in invalid return code testing or invalid return codes in messages.

operator
the type of comparison to be made with the return code. Operators and their meanings are:

GT...greater than
GE...greater than or equal to
EQ...equal to
LT...less than
LE...less than or equal to
NE...not equal to

stepname
the name of a preceding job step that issued the return code to be tested. When a one-level stepname is used in a procedure, the return code is effective only for steps that are contained in that procedure. The return code is not effective for steps outside the procedure or for previous executions of the same procedure.

stepname.procstepname
the "stepname"is the name assigned to the step requesting execution of a procedure and "procstepname" is the name of the step within the procedure that issued the return code to be tested.

EVEN
specifies that the job step is to be executed if any preceding job step has abnormally terminated and no return code tests are satisfied. If any return code tests are satisfied, this job step is bypassed. No return code tests, however, are made against steps that abnormally terminate. If EVEN or ONLY are not coded and a preceding job step fails, current job step return code tests will not be made and the step will be bypassed. If the termination is due to an operator CANCEL command, COND=EVEN is ignored.

ONLY
specifies that the job step is to be executed only if a preceding step has abnormally terminated and no return code tests are satisfied. If any return code tests, are satisfied, this job step is bypassed. No return code tests, however, are made against steps that abnormally terminate. If EVEN or ONLY are not coded and a preceding job step fails, current job step return code tests will not be made and the step will be bypassed.

General Rules for Coding

- Return code tests against preceding steps that do not run are ignored and the current step is executed.
- Return code tests are not made against a step that abnormally terminated and execution of the current step is determined by the use of the EVEN or ONLY parameter.
- If COND is coded on the JOB statement and on the EXEC statement, and the return code test requested on the JOB statement is satisfied, the job terminates even if the return code test requested on the EXEC statement is not satisfied. Likewise, if the return code test requested on the EXEC statement is satisfied, the step is bypassed even if the test requested on the JOB statement is not satisfied.
- If you code neither EVEN nor ONLY, you can make up to eight tests on return codes issued by preceding job steps or cataloged procedure steps that completed normally. If you code EVEN or ONLY, you can make up to seven tests on return codes. If you specify more than eight tests, a JCL error message is issued and the job fails.
- If you code only EVEN or ONLY, or if you code only one test, you need not code outer parentheses.
- If a job step that specifies the EVEN or ONLY subparameter refers to a data set that was to be created or cataloged in a preceding step, the data set may be incomplete if the step creating it abnormally terminated.
- You can code the EVEN or ONLY subparameters before, between, or after return code tests.
- If you want each return code test to be made on the return code issued by every preceding step, do not code a stepname or procstepname.
- If ONLY is specified on the first job step and a JOBLIB is being used, the unit and volume information are not passed to the succeeding step and the catalog will be searched for the JOBLIB data set.
- If a job step refers to a data set that was to have been created in a preceding step, that data set will not exist if the preceding step was bypassed. If a data set was cataloged in a preceding job step and you make a backward reference to that data set, unit and volume information for the data set will not be available if the preceding step was skipped.
- The COND parameter should not be specified in a deferred restart step or in any steps following the restart step if they contain values in the form stepname or stepname.procstepname that refer to a step preceding the restart step.
- Before resubmitting a job, all EXEC statements that contain the COND parameter should be carefully reviewed. If any of the COND parameters contain values referencing a step preceding the restart step, be aware that the COND parameters will be ignored for these steps.

Code COND on the EXEC statement for any of the following:

- When you want to specify different tests for each job step.
- If a test you specify is true, you want to skip just that one step, rather than bypassing all subsequent steps in the job.
- When you want to name a specific step whose return code is to be tested.
- When you want to specify special conditions for executing a job step.

When Coding to Run Under JES3

When JES3 determines the setup requirements for any given job, it does so without regard to any COND parameters specified on the EXEC statements of the job. All jobs are processed as though each step will be executed.

Examples of the COND Parameter

```
//STEP6     EXEC    PGM=BAB,COND=(4,GT,STEP3)
```

If 4 is greater than the return code issued by STEP3 (a return code of 0 to 3), the system will bypass the step. (A return code of 4 or greater from STEP3 allows this step (STEP6) to be executed.) Since neither EVEN nor ONLY is specified, this job step will be automatically bypassed if a preceding step abnormally terminates.

```
//TEST2     EXEC    PGM=BACK,COND=((16,GE),(90,LE,STEP1),ONLY)
```

If 16 is greater than or equal to the return code issued by any of the preceding job steps or if 90 is less than or equal to the return code issued by STEP1, this step will be bypassed. If none of the tests are satisfied and a preceding job step has abnormally terminated, this step will be executed because ONLY is coded.

```
//STP4      EXEC    BILLING,COND.PAID=((20,LT),EVEN),
//                  COND.LATE=(60,GT,FIND),
//                  COND.BILL=((20,GE),(30,LT,CHGE))
```

This example specifies that different return code tests pertain to each of the named procedure steps (PAID, LATE, and BILL). If the return code test specified for the procedure step named PAID is not satisfied, the step will be executed even if a preceding step abnormally terminates.

The DPRTY Parameter—*keyword, optional*

The DPRTY parameter assigns a dispatching priority to an address space. Dispatching priority will be used by the system resources manager in determining the order in which tasks will be executed.

For further information on the use of the DPRTY parameter, see "Assigning a Dispatching Priority to Job Steps" for either JES2 or JES3.

```
DPRTY[.procstepname]=([value1][,value2])
```

value1
a number from 0 through 15 which indicates whether the job step is to have the same priority or a different priority than the job. The job priority is coded on the JES2 PRIORITY statement or calculated from values specified in the accounting information on the JOB statement, the JOBPARM values, or an installation default.

value2
a number from 0 through 15 which the system adds to value1 to form the dispatching priority. The system forms the internal priority by converting the value assigned to value1 in the DPRTY parameter.

Default: If you omit the DPRTY parameter completely, the job step is assigned the APG priority. If value1 is omitted or it is equal to the APG value, the step is assigned the APG priority and any value you code for value2 is ignored. In this case, value2 is obtained from the Installation Performance Specification (IPS) using the performance group associated with the job step. (See **OS/VS2 System Programming Library: Initialization and Tuning Guide** for information on IPS.) If value2 is not specified in the IPS, a value of 6 is assigned to value2.

General Rules for Coding

- If you omit value1, you must code a comma preceding value2 to indicate the absence of value1.
- If you omit value2, you need not code the parentheses.

Cataloged Procedures

- You can code the DPRTY parameter on the EXEC statement of a cataloged procedure.
- If a job step calls a cataloged procedure:

To override all DPRTY parameters code the DPRTY parameter on the EXEC statement that calls the cataloged procedure. This will establish one dispatching priority for all steps in the procedure.

To override only certain DPRTY parameters code DPRTY[.procstepname] on the EXEC statement that calls the procedure for each procedure step that you want to override. The dispatching priority will then pertain only to the named procedure step.

Example of the DPRTY Parameter

```
//BP2       EXEC    PGM=FOUR,DPRTY=(13,9)
```

The system uses these numbers to form a dispatching priority for this step. Since the numbers are high, the dispatching priority will be high.

The DYNAMNBR Parameter—*keyword, optional*

The DYNAMNBR parameter specifies that a number of resources can be held in anticipation of reuse. DYNAMNBR can be coded instead of coding several DD DYNAM statements.

For further information on the DYNAMNBR parameter, see "Dynamically Allocating and Deallocating Data Sets."

```
DYNAMNBR[.procstepname]=n
```

n
specifies the number of DD DYNAM statements that you would otherwise have had to code; from 1 to 1635.

Default: 0

If DYNAMNBR is coded incorrectly, zero is assumed and a warning message is issued.

Rules for Coding

- The number of DD DYNAMs plus the value of the DYNAMNBR parameter cannot exceed 1635.
- If procstepname is omitted, DYNAMNBR applies to all steps in the procedure.
- The limit of the number of resources that can be held in anticipation of reuse is equal to the number of DD statements coded in the step plus the value specified on the DYNAMNBR parameter.

Example of the DYNAMNBR Parameter

```
//STEP1     EXEC    ACCT,DPRTY=(13,9),DYNAMNBR.CALC=12
```

This statement specifies that 12 allocations in the step CALC can be held in anticipation of reuse.

The PARM Parameter—*keyword, optional*

The PARM parameter passes variable information to a program in execution. For further information on the PARM parameter, see **OS/VS2 Supervisor Services and Macro Instructions.**

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

value
up to 100 characters of information which the system is to pass to a processing program.

General Rules for Coding

- Before coding the PARM parameter, see Figure 18, "Character Sets" for an explanation of alphameric, national, and special characters.
- If the value contains more than one expression separated by commas, you must enclose it in apostrophes or parentheses; that is, PARM='P1,123,MT5' or PARM=(P1,123,MT5). Enclosing apostrophes and parentheses are not passed to the processing program; commas within apostrophes and parentheses are passed as part of the value.
- If you include special characters in any expression, either (1) enclose the value in apostrophes, or (2) enclose the expression in apostrophes and the value in parentheses; that is, PARM='P50,12+80' or PARM=(P50,'12+80'). Apostrophes or parenthesis used to enclose values are not passed as part of the value; apostrophes used to enclose expressions are passed as part of the expression.
  If one of the special characters is an ampersand and you are not defining a symbolic parameter, code two consecutive ampersands in its place; that is, PARM='3462& &5'. When two apostrophes or two ampersands are coded, only one is passed to the processing program.
- If you must continue the value on another statement, enclose it in parentheses. The continuation comma is considered part of the value field and counts toward the maximum of 100 characters of data. Any expression that contains special characters must be enclosed in apostrophes and cannot be continued on the next statement.

Calling Cataloged or In-stream Procedures

- If a job step calls a cataloged or in-stream procedure, you can pass information to the first procedure step and nullify all other PARM parameters in the procedure or override some of the PARM parameters contained in the procedure.
- To nullify the PARM parameters in the procedure, code the PARM parameter on the EXEC statement that calls the procedure. The information contained in the PARM parameter is passed to the first procedure step and PARM parameters in all other procedure steps are nullified.
- To override some of the PARM parameters contained in the procedure, code, on the EXEC statement that calls the procedure, PARM.procstepname for each procedure step that you want to override. Information provided in the PARM value is passed only to the named procedure step.

Examples of the PARM Parameter

```
//RUN3      EXEC    PGM=APG22,PARM='P1,123,P2=5'
```

Passes the information in the PARM parameter, except the apostrophes, to the processing program named APG22.

```
//          EXEC    PROC81,PARM=MT5
```

Passes this information to the first step of the procedure named PROC81. If any of the other procedure steps in PROC81 contain the PARM parameter, those parameters are nullified.

```
//STP6      EXEC    ASMFCLG,PARM.LKED=(MAP,LET)
```

Passes this information to the procedure step named LKED. If any of the other procedure steps in ASMFCLG contain the PARM parameter, those parameters are still in effect.

```
//RUN4      EXEC    PGM=IFOX00,PARM=(NOOBJECT,'LINECNT=50',
//          DECK)
```

Passes the information in the PARM parameter, except the apostrophes and the parentheses, to the processing program named IFOX00. When specifying additional PARM expressions on a continuation card, the entire value must be enclosed in parentheses. Any expression within apostrophes must be contained on a single statement.

The PERFORM Parameter—*keyword, optional*

The PERFORM parameter specifies the performance group definitions to which a job step is associated.

For further information on the performance groups, see "Performance of Jobs and Jobs Steps."

```
PERFORM[.procstepname]=n
```

n
is a number between 1 and 999 inclusive, identifying a performance group that has been defined by the installation.

Default: The interpreter obtains a default from the system resources manager and issues a warning message indicating a system default is set.

- For non-TSO jobs, the IBM-supplied default is one (1).
- For TSO jobs, the IBM-supplied default is two (2).
- An installation-supplied or IBM-provided default can override the performance group number you code if it fails to pass validity checks. The **OS/VS2 SPL: Initialization and Tuning Guide** contains a detailed description of the override procedure.

Note: Consult your systems programmer to find out which performance groups are valid and which performance groups apply to which job types.

Rules for Coding

- If PERFORM is specified for a procedure, the specified value is effective for the entire procedure. If PERFORM.procstepname is coded for a procedure, the value is effective only for the procedure step named.
- If PERFORM is specified on the JOB statement, its value supercedes any PERFORM specifications on EXEC statements associated with the job.
- If no PERFORM parameter appears on the JOB statement and a PERFORM parameter appears on an associated EXEC statement, the parameter value appearing on the EXEC statement is used during the associated job step.

Example of the PERFORM Parameter

```
//STEPA     EXEC     PGM=ADAM,PERFORM=60
```

This job step will be run in performance group 60 if it passes validity checks. The significance of this performance group is defined by the installation.

The PGM Parameter—*positional, optional*

The PGM parameter specifies a program to be executed. The specified program must be a member of a temporary library, a system library, or a private library.

For further information on identifying programs and on libraries (partitioned data sets), see "Identifying Data Sets to the System."

```
PGM=      {program name                        }
          {*.stepname.ddname                   }
          {*.stepname.procstepname.ddname      }
```

program name
specifies the member name or alias of the program to be executed.

***.stepname.ddname**
specifies a backward reference to a DD statement that defines, as a member of a partitioned data set, the program to be executed. Stepname is the name of the step in which the DD statement appears; ddname is the name that appears on the DD statement.
This form of the parameter is usually used when a previous job step has created a temporary partitioned data set to store a program until the program is required.

***.stepname.procstepname.ddname**
specifies a backward reference to a DD statement within a cataloged procedure step that defines, as a member of a partitioned data set, the program to be executed. Stepname is the name of the step that calls the procedure; procstepname is the name of the procedure step that contains the DD statement .

Rules for Coding

- If you code the PGM parameter, code it as the first parameter on the EXEC statement. The program you specify must be a member of a partitioned data set.
- The PGM and PROC parameters are mutually exclusive; therefore, if you code the PGM parameter, do not code the PROC parameter.
- The program name must consist of up to 8 alphameric or national characters, the first of which must be alphabetic or national.
- When running under JES3, if you code PGM=JCLTEST or PGM=JSTTEST, you can scan the job's JCL without executing the job or setting up devices. This is the same function as coding TYPRUN=SCAN on the JOB statement.

Examples of the PGM Parameter

```
//STEP1     EXEC    PGM=TABULATE
```

Specifies that the program named TABULATE is a member of SYS1.LINKLIB.

```
//JOB8      JOB     MSGLEVEL=(2,0)
//JOBLIB    DD      DSNAME=DEPT12.LIB4,DISP=(OLD,PASS)
//STEP1     EXEC    PGM=USCAN
```

Specifies that the system is to look for the program name USCAN in a private library named DEPT12.LIB4.

```
//CREATE    EXEC    PGM=IEWL
//SYSLMOD   DD      DSNAME=&&PARTDS(PROG),UNIT=2314,DISP=(MOD,PASS),
                    SPACE=(1024,(50,20,1))
//EXECUTE   EXEC    PGM=*.CREATE.SYSLMOD
```

Uses a backward reference to a DD statement that defines a temporary library created in the step named CREATE. The program name PROG is stored as a member of the partitioned data set named &&PARTDS and will be executed in the step name EXECUTE. The data set &&PARTDS will be deleted at the end of the job.

```
//STEP2     EXEC    PGM=UPDT
//DDA       DD      DSNAME=SYS1.LINKLIB(P40),DISP=OLD
//STEP3     EXEC    PGM=*.STEP2.DDA
```

Uses a backward reference to a DD statement that defines a system library. The program named P40 is stored as a member of SYS1.LINKLIB and is executed in the step named STEP3.

```
//CHECK     EXEC    PGM=IEFBR14
```

Executes the program named IEFBR14, allowing you to satisfy space allocation and disposition processing requests prior to executing your program. T he remaining job control statements in the job are also checked for syntax.

The PROC and Procedure Name Parameters—*positional, optional*

The PROC parameter specifies that a cataloged procedure or an in-stream procedure is to be called and executed.

For further information on cataloged and in-stream procedures, see "Using Cataloged and In-Stream Procedures."

```
{PROC=procedure name}
{procedure name     }
```

procedure name
the member name (or alias) of a cataloged procedure or the name of an in-stream procedure to be called and executed.

Rules for Coding

- The procedure name must consist of one through eight alphameric or national characters of which the first must be alphabetic or national.
- The PROC and PGM parameters are mutually exclusive. Therefore, if you code PGM, do not code PROC. If you code the PROC parameter, code it as the first parameter on the EXEC statement, instead of the PGM parameter. You can code only the cataloged or in-stream procedure name, omitting PROC.
- When the EXEC statement specifies a cataloged or in-stream procedure, parameters in the operand field of the EXEC statement will override EXEC parameters in the called procedure.
- Any DD statements that follow the EXEC statement will be treated as overriding DD statements or DD statements that are to be added to the cataloged or in-stream procedure for the duration of the job step.

Examples of the PROC Parameter

```
//SP3        EXEC    PROC=PAYWKRS
```

Specifies that the cataloged or in-stream procedure named PAYWKRS is to be called.

```
//BK         EXEC    OPERATE
```

Specifies that the cataloged or in-stream procedure named OPERATE is to be called. This specification has the same effect as coding PROC=OPERATE.

The RD Parameter—*keyword, optional*

The RD (restart definition) parameter specifies how the step restart facilities are used with the CHKPT macro instruction, and whether automatic restart is permitted or suppressed.

For detailed information on the checkpoint/restart facilities, refer to **OS/VS2 MVS Checkpoint/Restart.**

```
RD[.procstepname]= {R   }
                   {RNC }
                   {NC  }
                   {NR  }
```

R
indicates that automatic step restart is permitted.
If the processing program used by a job step does not include any CHKPT macro instructions, coding RD=R allows execution to be resumed at the beginning of the abnormally terminated step.
If the program does include a CHKPT macro instruction, coding RD=R permits automatic step restart to occur if the step abnormally terminates before execution of the CHKPT macro instruction; thereafter, only checkpoint restart can occur.
If you cancel the effects of the CHKPT macro instruction before a checkpoint restart is performed, the request for automatic step restart is again in effect.

RNC
indicates that automatic step restart is permitted and automatic and deferred checkpoint restart are not permitted.

NC
indicates that automatic step restart and automatic and deferred checkpoint restart are not permitted.

NR
indicates that a CHKPT macro instruction can establish a checkpoint, but neither automatic step restart nor automatic checkpoint restart is permitted if the job fails. (If you have coded RD=NR and a system failure occurs, RD=NR does not prevent the job from restarting.) Coding RD=NR allows you to resubmit the job at a later time and specify in the RESTART parameter (on the job statement of the resubmitted job) the checkpoint at which execution is to be resumed.

Rules for Coding

- Automatic restart will not be honored if you do not have a job journal. The journal data set in JES3 is used if one of the following exists:
  - RD= is specified on the JOB or EXEC statement.
  - JOURNAL=YES is specified on the JES3 MAIN statement.
  - JOURNAL=YES is specified on the CLASS initialization statement for this job, and was not overridden by the JES3 MAIN statement.
- If you specify RD=R or RD=RNC on either the JOB or EXEC statements, JES2 forces journaling.
- Even if you do not code the RD parameter, your job is eligible for automatic checkpoint restart if checkpoints have been issued. However, the job is not eligible for automatic step restart.

- Code the RD parameter on EXEC statements instead of the JOB statement when you want to make different restart requests for each job step.
- If you code the RD parameter on the JOB statement, any RD parameters coded on the job's EXEC statements are ignored and the value coded on the JOB statement is effective for all steps.
- The RD parameter can be coded on the EXEC statement of a cataloged procedure. If the job step does call a cataloged procedure:
- To override all RD parameters, code the RD parameter on the EXEC statement that calls the procedure. This establishes one restart for all the steps in the procedure.
- To override only certain RD parameters, code, on the EXEC statement that calls the procedure, RD.procstepname for each procedure step that you want to override. The restart request will then pertain only to the named procedure step.
- The RD parameter applies to the step corresponding to the statement or to all steps of the cataloged procedure referred to by the statement.
- RD.procstepname applies to the specified procedure step and overrides the RD parameter that may be coded on the EXEC statement of the procedure step. It can be coded once for each step of the procedure.
- The RD parameter values NC and RNC can be used to suppress the action of the CHKPT parameter.

Examples of the RD Parameter

```
//STEP1     EXEC    PGM=GIIM,RD=R
```

Permits execution to be automatically restarted with this step if it abnormally terminates.

```
//NEST      EXEC    PGM=T18,RD=RNC
```

Permits execution to be automatically restarted with this step if it abnormally terminates; suppresses the action of CHKPT macro instructions issued in the program this job step uses.

```
//CARD      EXEC    PGM=WTE,RD=NR
```

Neither automatic step restart nor automatic checkpoint restart can occur, but CHKPT macro instructions issued in the program that this job step executes can establish checkpoints.

```
//STP4      EXEC    BILLING,RD.PAID=NC,RD.BILL=NR
```

Specifies that different restart requests pertain to each of the named procedure steps (PAID and BILL).

The REGION Parameter—*keyword, optional*

The REGION parameter specifies the amount of space to be allocated to a job step.

For further information on the REGION parameter, see the section "Requesting Storage For Execution of a Program."

```
REGION[.procstepname]=valueK
```

valueK
specifies a number that indicates how many bytes of storage are to be allocated to a job step.

Default: the installation-defined default.

General Rules for Coding

- Code an even number. (If you code an odd number, the system treats it as the next highest even number.)
- When you want to specify a different region size for each job step, code the REGION parameter on the EXEC statements, instead of the JOB statement.
- If the REGION parameter is coded on the JOB statement, REGION parameters coded on the job's EXEC statements are ignored.
- If you code the region value as 0, or allow it to default to 0, results are unpredictable.

Rule for Coding when Using Virtual Storage

The installation provides a default region size for use when the REGION parameter is not specified when ADDRSPC=VIRT is code or implied. That value, or the value specified on the REGION parameter if coded, sets the upper boundary to limit region size for variable-length GETMAINs as long as there remains in the region at least the minimum amount of storage requested. In addition, the region value is used by an IBM or installation-supplied routine (IEALIMIT) to establish a second value used to limit fixed-length GETMAINs, and variable-length GETMAINs when the space remaining in the region is less than the requested minimum. When the minimum requested length for variable-length GETMAINs causes this second value to be exceeded, the job or job step abnormally terminates. For further information, see **OS/VS2 System Programming Library: Supervisor.**

Examples of the REGION Parameter

```
//MKBOYLE   EXEC    A,ADDRSPC=REAL,REGION=40K
```

40K of real storage is assigned to this job step.

```
//STP6      EXEC    PGM=CONT,REGION=120K
```

A 120K region is assigned. For variable-length GETMAINs, the request is satisfied as long as there remains in the region at least the minimum amount requested. If less than the minimum remains, the minimum is allocated as long as the allocation does not cause the second limiting value (described above) to be exceeded. For fixed-length GETMAINs, the amount requested is allocated as long as the allocation does not cause the second limiting value to be exceeded.

The TIME Parameter—*keyword, optional*

The TIME parameter specifies the maximum amount of time that a job step can use the CPU. The CPU time used is written on the output listing.

```
TIME[.procstepname]=     ([minutes][,seconds])
        1440
```

minutes
specifies the maximum number of minutes the job step can use the CPU. The number of minutes must be less than 1440 (24 hours).

seconds
specifies the maximum number of seconds beyond the specified number of minutes the job step can use the CPU. The number of seconds must be less than 60.

1440
specifies the job can use the CPU for an unlimited amount of time. Note that job step accounting is still performed for TIME=1440 users.

Rules for Coding

- If you code the CPU time limit in minutes only, you need not code the parentheses.
- Code 1440 if the job step can use the CPU for 24 hours or more or if the job step should be allowed to remain in a wait state for more than the established time limit.
- You can code the TIME parameter on the EXEC statement of a cataloged procedure step. To override all TIME parameters in a cataloged procedure, code the TIME parameter on the EXEC statement that calls the procedure. This establishes a CPU time limit for the entire procedure, and nullifies any TIME parameters that appear on EXEC statements in the procedure.
  To override only certain TIME parameters, code, on the EXEC statement that calls the procedure, TIME.procstepname, for each procedure step that you want to override. The CPU time limit will then pertain only to the named procedure step.
- The remaining job time may affect the amount of time the step can use the CPU. If the remaining CPU time for the job is less than the CPU time limit specified on the EXEC statement, the step can use the CPU only for the job's remaining CPU time.
- Because the CPU time-used field is checked at intervals of about 10.5 seconds, the actual amount of time that a job step uses the CPU can exceed the time specified on the TIME parameter by up to 10.5 seconds.
- A step that exceeds the specified limit causes termination of the job unless you use a user exit routine to extend the time.
- When TIME=0 is coded on the EXEC statement, the step will fail only after the unexpired time from the previous step is used up.
- The time limit specified for a step is rounded to the nearest unit (1 unit = 1.048576 seconds).

Examples of the TIME Parameter

```
//STEP1     EXEC    PGM=GRYS,TIME=(12,10)
```

Specifies that the maximum amount of time the step can use the CPU is 12 minutes 10 seconds.

```
//FOUR      EXEC    PGM=JPLUS,TIME=(,30)
```

Specifies that the maximum amount of time the step can use the CPU is 30 seconds.

```
//INT       EXEC    PGM=CALC,TIME=5
```

Specifies that the maximum amount of time the step can use the CPU is 5 minutes.

```
//LONG      EXEC    PGM=INVANL,TIME=1440
```

Specifies that the step can have unlimited use of the CPU. Therefore, the step can use the CPU and can remain in a wait state for an unspecified period of time if not restricted by the TIME parameter specified on the JOB statement.

```
//STP4      EXEC    BILLING,TIME.PAID=(45,30),TIME.BILL=(112,59)
```

Specifies that different time limits pertain to each of the named procedure steps.

Refer to "The TIME Parameter" under "The Job Statement" for examples of coding TIME on both the JOB and EXEC statements.

Control Statement

The DD (data definition) statement describes a data set to be used in a job step and specifies the input and output facilities required for the data set.

```
//ddname    DD  operands      comments
```

The DD statement consists of the characters // in columns 1 and 2, and four fields — the name, operation (DD), operand, and comments field.

General Rules for Coding

- Code a DD statement for each data set to be used in a step.
- Code a ddname, beginning in column 3, and consisting of 1 through 8 alphameric and national (@,#,$) characters. The first character must be alphabetic or national.
- Code unique ddnames within a job step. If duplicate ddnames exist in a job step, processing is as follows:
  - Under JES2,
    - Allocation of devices, space, and disposition processing are done for both DD statements; however, all references are directed to the first such DD statement in the step.
  - Under JES3,
    - If both DD statements request JES3, or jointly managed devices, the job is cancelled during the JES3 interpreter phase.
    - If only one or if neither DD statement requests JES3 or jointly managed devices, the allocation of devices, space and disposition processing are performed for both DD statements; however, all references are directed to the first such DD statement in the step.
- Apart from the restricted use of certain special ddnames (listed below), there are two instances when you should not code a ddname at all:
  - A DD statement is to define a data set that is concatenated with a data set defined by a preceding DD statement.
  - The DD statement is the second or third consecutive DD statement that defines an indexed sequential data set.
- Special ddnames: Do not use the following nine special ddnames unless you wish to make use of the particular facilities which these names represent to the system; these facilities are explained in detail in the following pages.

JOBCAT	SYSMDUMP
JOBLIB	SYSUDUMP
STEPCAT	SYSCHK
STEPLIB	SYSCKEOV
SYSABEND	

Under JES3, the following ddnames are reserved and should not be used on a DD statement:

JCBIN	JCLIN	JESMSG
JCBLOCK	JESInnnn	JOURNAL
JCBTAB	JESJCL	SYSMSG

If you have installed MVS/System Product Release 2 (5740-XYN or 5740-XYS) without installing Data Facility/Device Support Release 1 Enhancements (5740-AM7) do not use the checkpoint/restart facility.

If you have installed Data Facility/Device Support with either of the MVS/System Products you can use the checkpoint/restart facility with certain restrictions.

For additional information concerning these restrictions see **OS/VS2 MVS Checkpoint/Restart** and **OS/VS2 MVS System Programming Library: Job Management.**

- Although all DD statement parameters are optional, a blank operand field is invalid, except when you are overriding DD statements that define concatenated data sets.
- The maximum number of DD statements allowed per job step is 1635.
- JES3 allows the installation to determine the maximum number of DD statements that you can code.
- There are two types of parameters that can be coded on a DD statement: keyword and positional. The positional parameters, which must precede any keyword parameters, are:

  *
  DATA
  DUMMY
  DYNAM

The keyword parameters are:

AMP	DEST	FREE	QNAME	VOLUME
BURST	DISP	HOLD	SPACE	
CHARS	DLM	LABEL	SUBSYS	
CHKPT	DSID	MODIFY	SYSOUT	
COPIES	DSNAME	MSVGP	TERM	
DCB	FCB	OUTLIM	UCS	
DDNAME	FLASH	PROTECT	UNIT	

Rules for Coding when Using Cataloged Procedures

- If a job step uses a cataloged procedure, you can make modifications to the DD information within the procedure for the duration of the job step. To do this, code modifications on the DD statements immediately following the EXEC statement that calls the cataloged procedure.
- To override parameters on a DD statement within a cataloged procedure, code the name of the procedure step in which the DD statement appears, followed by a period, followed by the name of the DD statement that you want to override.
- To override two or more DD statements in a procedure step, the sequence of the overriding statements must be the same as the sequence of the procedure statements being overridden.
- To add DD statements to a procedure step, code the name of the procedure step in which the DD statement appears, followed by a period, followed by a ddname of your choosing. Added statements must follow all overriding statements.
- To supply a procedure step with data in the input stream, code the name of the procedure step that is to use the data, followed by a ddname. This ddname may be predefined in the procedure step by means of the DDNAME parameter. In this case, the ddname that follows the procedure step name must be the name code in the DDNAME parameter.
- Do not use a JOBLIB DD statement in a cataloged procedure.

Examples of Valid Ddnames

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

Because the ddname is missing from the second DD statement, the data sets defined in these statements will be concatenated.

```
//PAYROLL.DAY   DD
```

If the procedure step named PAYROLL includes a DD statement named DAY, this statement will override parameters on the statement named DAY. If the step does not include a DD statement named DAY, this statement will be added to the procedure step for the duration of the job step.

```
//STEPSIX.DD4   DD
//              DD
```

This sequence defines data sets that are to be concatenated and added to the procedure step. On the first DD statement, the procedure step to which statements are to be added is identified and followed by any valid ddname. On the second DD statement, the ddname is omitted.

The JOBCAT Facility

DD statement

The JOBCAT DD statement defines a private user catalog for the duration of a job. The system searches the private user catalog for data sets before it searches the master catalog or a private catalog associated with the first qualifier of a data set's name.

```
//JOBCAT    DD      DISP= {OLD},DSNAME=usercatalogname
                        {SHR}
```


Rules for Coding

- JOBCAT applies to each job step in which a STEPCAT has not been specified.
- The location of the user catalog is given in the master catalog, so do not specify any unit or volume information.
- To specify more than one user catalog for a job, include after the JOBCAT statement an unlabeled DD statement that names another user catalog.
- OS CVOLs cannot be specified as JOBCAT. Access to an OS CVOL is only possible with a CVOL pointer in the master catalog.
- The JOBCAT statement must appear after the JOB statement, but before the first EXEC statement.
- A JOBLIB statement must precede the JOBCAT statement, if specified.

Example of the JOBCAT DD Statement

The following example specifies a private user catalog. Place a JOBCAT DD statement before the first EXEC statement after the JOB statement. The JOBCAT DD statement should also appear after any JOBLIB statements.

```
//EXAMPLE   JOB     WILLIAMS,MSGLEVEL=1
//JOBLIB    DD      DSNAME=USER.LIB,DISP=SHR
//JOBCAT    DD      DSNAME=LYLE,DISP=SHR
//          EXEC    PGM=SCAN
```

The JOBLIB Facility

DD statement

The JOBLIB DD statement defines a private library to be searched, prior to the system libraries, for the program named in the PGM= field of the EXEC statement.

```
//JOBLIB     DD
```

General Rules for Coding

- Code JOBLIB as the ddname on the first DD statement. Never use the ddname JOBLIB except to define a private library for an entire job.
- Omit the ddname from all subsequent DD statements that define additional private libraries. These DD statements must immediately follow the JOBLIB statement. The JOBLIB statement must immediately follow the JOB statement and any JES statements. There must be no intervening EXEC or other DD statements between the JOBLIB statement and the JOB card.
- If you include a JOBLIB DD statement in the JCL for a job, each time the job requests a program, the system first searches the private library; if it does not find the program there, it next searches the system library.
- Use a STEPLIB DD statement, described under the STEPLIB Facility, to define a private library to be made available to 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 job library is ignored for that step.
- To make the private library available throughout the job, code the DISP parameter to specify the library's status and disposition. One of the following combination of DISP parameter values must be coded:

  ```
  DISP=(NEW,PASS)
  DISP=(OLD,PASS)
  DISP=(SHR,PASS)
  DISP=(NEW,CATLG)
  DISP=(OLD,CATLG)
  DISP=(SHR,CATLG)
  ```

  For further explanation, see the discussion of the DISP parameter.

- The rules for coding parameters on the JOBLIB DD statement depend on whether or not the private library is cataloged. These rules are discussed below under separate headings.
- Do not use a JOBLIB statement in a cataloged procedure.
- If COND=ONLY is specified on the first job step and a JOBLIB is being used, the unit and volume information are not passed to the succeeding step and the catalog will be searched for the JOBLIB data set.

Rules for Coding When the Library is Cataloged

- Code the DSNAME parameter to specify the name of the private library.
- Code the DISP parameter. The DISP parameter must be other than NEW.
- Do not code VOL=SER to request a cataloged data set.
- Code the DCB parameter if complete data control block information is not contained in the data set label.
- To refer to the private library in a later DD statement, code DSNAME=*.JOBLIB and the DISP parameter.

- If a later DD statement defines a data set that is to be placed on the same volume as the private library, code VOLUME=REF=*.JOBLIB to obtain volume and unit information.

Rules for Coding When the Library is Not Cataloged

- Code the DISP parameter. The DISP parameter must be one of the following values:

  ```
  DISP=(OLD,PASS)
  DISP=(SHR,PASS)
  DISP=(NEW,PASS)
  ```

- Code the UNIT parameter to specify the device to be allocated to the library.
- Code the DSNAME parameter unless the data set has been assigned a disposition of (NEW,PASS).
- Code the VOLUME parameter unless the status of the data set is NEW.
- If the status of the data set is NEW, you must code the SPACE parameter to allocate space for the data set on the designated volume.
- Code the DCB parameter if complete data control block information is not contained in the data set label or in the problem program.
- To refer to the private library in a later DD statement, code DSNAME=*.JOBLIB,VOLUME=REF=*.JOBLIB (or VOLUME=SER=serial number, UNIT=unit information), and the DISP parameter, DISP=(OLD,PASS).
- If a later DD statement defines a data set that is to be placed on the same volume as the private library, code VOLUME=REF=*.JOBLIB to obtain volume and unit information.

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 defined on the JOBLIB DD statement is cataloged. The statement named DD1 refers to the private library defined in the JOBLIB DD statement.

```
//PAYROLL   JOB     FOWLER,CLASS=L
//JOBLIB    DD      DSNAME=PRIV.DEPT58,DISP=(OLD,PASS),
//                  UNIT=2314,VOLUME=SER=D58PVL
//STEP      EXEC    PGM=DAY
//STEP2     EXEC    PGM=BENEFITS
//DD1       DD      DSNAME=*.JOBLIB,VOLUME=REF=*.JOBLIB,
//                  DISP=(OLD,PASS)
```

The private library defined on the JOBLIB DD statement is not cataloged. The statement named DD1 refers to the private library defined in the JOBLIB DD statement.

```
//TYPE      JOB     MSGLEVEL=(1,1)
//JOBLIB    DD      DSNAME=GROUP8.LEVEL5,DISP=(NEW,CATLG),
//                  UNIT=2314,VOLUME=SER=148562,
//                  SPACE=(CYL,(50,3,4))
//STEP1     EXEC    PGM=DISC
//DDA       DD      DSNAME=GROUP8.LEVEL5(RATE),DISP=MOD,
//                  VOL=REF=*.JOBLIB
//STEP2     EXEC    PGM=RATE
```

The private library defined on the JOBLIB DD statement does not exist yet; therefore, all the parameters required to define the private library are included on the JOBLIB DD statement. The library is not created until STEP1 when a new member is defined for the library. The system looks for the program named DISC in SYS1.LINKLIB; the system first looks for the program named RATE in the private library.

```
//PAYROLL   JOB     LIEF,TIME=1440
//JOBLIB    DD      DSNAME=KRG.LIB12,DISP=(OLD,PASS)
//          DD      DSNAME=GROUP31.TEST,DISP=(OLD,PASS)
//          DD      DSNAME=PGMSLIB,UNIT=2314,
//                  DISP=(OLD,PASS),VOLUME=SER=34568
```

Several private libraries are concatenated. The system searches libraries for each program in this order: KRG.LIB12, GROUP31.TEST, PGMSLIB, SYS1.LINKLIB.

The STEPCAT Facility

DD statement

The STEPCAT DD statement defines a private VSAM user catalog for the duration of a job step. The system searches the private user catalog for data sets before it searches the master catalog or a private catalog associated with the first qualifier of a data set's name. s

```
//STEPCAT   DD      DISP= {OLD},DSNAME=usercatalogname
                        {SHR}
```


Rules for Coding

- A STEPCAT DD statement can appear in any position among the DD statements for a job step.
- The location of the user catalog is given in the master catalog, so do not specify any unit or volume information.
- To specify more than one user catalog for a job step, include after the STEPCAT statement an unlabeled DD statement that names another user catalog.
- If you want to override the JOBCAT with the master catalog for a particular job step, code the following:

```
//STEPCAT       DD      DISP=OLD,DSNAME=master catalog name
```

- OS CVOLs cannot be specified as STEPCAT. Access to an OS CVOL is only possible with a special CVOL pointer in the master catalog.

Example of the STEPCAT DD Statement

The following example specifies a job-step user catalog named BETTGER by placing a DD statement with the ddname STEPCAT after the EXEC statement for the job step:

```
//          EXEC    PROC=SNZ12
//STEPCAT   DD      DSNAME=BETTGER,DISP=SHR
```

The STEPLIB Facility

DD Statement

The STEPLIB DD statement defines a private library to be made available by the system to a job step. This library is searched prior to the system libraries for the program named in the PGM= field of the EXEC statement.

Use a JOBLIB DD statement, described under "The JOBLIB Facility", to define a private library to be made available to an entire job. If you include a JOBLIB DD statement for the entire 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 requested program. The job library is ignored for that step.

```
//STEPLIB   DD
```

General Rules for Coding

- The ddname on this statement must be STEPLIB. Never use the ddname STEPLIB except to define a private library for a job step.
- A STEPLIB DD statement can appear in any position among the DD statements for a step.
- A private library defined on a STEPLIB DD statement can be referred to by, or passed on to, later job steps in the same job.
- If you include a STEPLIB DD statement in the JCL for a job, when the jobstep for which the library is defined requests the program, the system first searches the private library; if it does not find the program there, it next searches the system library.
- A STEPLIB DD statement can appear in a cataloged procedure.
- To concatenate libraries, that is, to arrange a sequence of DD statements that define different data sets:

  Code STEPLIB as the ddname of the first DD statement.

  Omit the ddname from all subsequent DD statements that define private libraries for the particular step.

- If you want the system to ignore the JOBLIB for a particular job step, use the following STEPLIB DD statement:

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

  For the particular job step, the system will first search the system library for the required data set.
- The rules for coding parameters on the STEPLIB DD statement depend on whether the library is cataloged, not cataloged, or passed by a previous job step. These rules are discussed below under separate headings.

Rules for Coding When the Library is Cataloged

- Code the DSNAME parameter to specify the name of the private library.
- Code the DISP parameter to specify the library's status and its disposition. Its status must be either OLD or SHR. Its disposition may be any valid disposition.
- Code the DCB parameter if complete data control block information is not contained in the data set label.

Rules for Coding When the Library has been Passed by a Previous Step

- Within a job, a previously defined step library must be made available for use by subsequent job steps by assigning a disposition of PASS.
- To refer to a previously defined step library:
  Code the DSNAME parameter, specifying either the name of the step library or a backward reference of the form *.stepname.ddname. If the step library was defined in a cataloged procedure, the backward reference must include the procedure step name *.stepname.procstepname.ddname.
  Code the DISP parameter, specifying a status of OLD and a disposition, depending on what you want done with the private library after its use in the job step.
  Code the DCB parameter if complete data control block information is not contained in the data set label.

Rules for Coding When the Library is Neither Cataloged Nor Passed

- Code the DSNAME parameter, specifying the name of the private library.
- Code the DISP parameter, specifying the library's status, either OLD or SHR and a disposition, depending on what you want done with the private library after its use in the job step.
- Code the VOLUME parameter, identifying the volume serial number.
- Code the UNIT parameter, specifying the device to be allocated to the library.
- Code the DCB parameter if complete data control block information is not contained in the data set label.

Examples of the STEPLIB DD Statement

```
//PAYROLL   JOB     BROWN,MSGLEVEL=1
//STEP1     EXEC    LAB14
//STEP2     EXEC    PGM=SPKCH
//STEPLIB   DD      DSNAME=PRIV.LIB5,DISP=(OLD,KEEP)
//STEP3     EXEC    PGM=TIL80
//STEPLIB   DD      DSNAME=PRIV.LIB13,DISP=(OLD,KEEP)
```

The private libraries defined in STEP2 and STEP3 are cataloged.

```
//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=2314,VOLUME=SER=55566
//STEP3     EXEC    PGM=A1530
//STEP4     EXEC    PGM=SNAP11
//STEPLIB   DD      DSNAME=*.STEP2.STEPLIB,
//                  DISP=(OLD,KEEP)
```

The private library defined in STEP2 is not cataloged. The STEPLIB DD statement in STEP4 refers to the library defined in STEP2. Since a JOBLIB DD statement is included, STEP1 and STEP3 could execute programs from LIB5.GROUP4 or, if the programs are not found there, from SYS1.LINKLIB. STEP2 and STEP4 could execute programs from LIBRARYP or 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=2314,VOLUME=SER=44455
//          DD      DSNAME=LIB.DEPT88,DISP=(OLD,KEEP)
//STEP4     EXEC    PGM=SHORE
```

STEP2 and STEP4 can use programs contained in the private library named LIB5.GROUP4, which is defined in the JOBLIB DD statement. STEP1 can use a program from the system library, since the library defined on the STEPLIB DD statement is 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 SYSABEND, SYSMDUMP, and SYSUDUMP Facilities

DD Statements

The SYSABEND DD statement defines a data set on which a formatted dump can be written if the step in which the statement appears abnormally terminates. The default dump provided by this facility includes the processing program storage area only.

The SYSMDUMP DD statement defines a data set on which an unformatted, machine-readable, dump can be written if the step in which the statement appears abnormally terminates. The default dump provided by this facility includes the system nucleus and the processing program storage area.

The SYSUDUMP DD statement defines a data set on which a formatted dump can be written if the step in which the statement appears abnormally terminates. The default dump provided by this facility includes only the processing program storage area.

For detailed information on the IBM-supplied defaults, see **OS/VS2 MVS System Programming Library: System Initialization and Tuning.**

For information on how to interpret dumps, see **OS/VS2 System Programming Library: Debugging Handbook** and **OS/VS2 System Programming Library: Diagnostic Techniques.**

```
//SYSABEND  DD
//SYSMDUMP  DD
//SYSUDUMP  DD
```

Rules for Coding

- If more than one of these DD statements is present, the system uses the last one it encounters. If you override one of these DD statements in a cataloged procedure, the overriding DD statement is used because it is the last one encountered.

To store a dump:

- If you wish to store a dump instead of having it printed, code the following parameters:
  - The DSNAME parameter, specifying the name of the data set.
  - The UNIT parameter, specifying the device to be allocated to the data set.
  - The VOLUME parameter, specifying the serial number of the volume to which the dump is to be written.
  - The DISP parameter, specifying the data set's status and disposition. Since you want to store the data set, make the data set's conditional disposition KEEP or CATLG.
  - The SPACE parameter (for direct access devices), specifying the amount of space you want allocated to the data set.
- Dumps written to a data set defined by a SYSMDUMP DD statement must be allocated to either a magnetic tape unit or a direct access device. If the job or step is running with nonpageable virtual storage (ADDRSPC=REAL on the JOB or EXEC statements), the SYSMDUMP output must be directed to a VIO data set.

- To take subsequent SYSMDUMPs to the same data set you must specify the following:
  - DSNAME=SYS1.SYSMDPxx (where xx can be 00 - FF) which is a preallocated data set.
  - DISP=SHR
- Subsequent dumps will not be taken if the previous dump is not off-loaded and an end-of-file mark written at the beginning of the SYS1.SYSMDPxx data set.

Note: Your installation must have written an exit routine for message IEA993 that will:

- Off-load the dump from the SYS1.SYSMDPxx data set
  and
- Write an end-of-file mark at the beginning of the SYS1.SYSMDPxx data set whenever the dump is off-loaded.

For further information on this exit routine see **OS/VS2 System Programming Library: Supervisor.**

To print a dump:

- To print a dump taken for either a SYSABEND or SYSUDUMP DD statement, code the UNIT parameter, specifying the unit record device to which you want to write the dump, or code the SYSOUT parameter, specifying the output class through which you want the data set routed.
- To print a dump taken for a SYSMDUMP DD statement, use the AMDPRDMP service aid described in **OS/VS2 System Programming Library: Service Aids,** or IPCS which is documented in **OS/VS2 MVS Interactive Problem Control System (IPCS) System Information.**
  - When using the AMDPRDMP service aid, the SYSMDUMP DD statement must allocate the dump data set to either a magnetic tape unit or a direct access device.

    If the job or job step is running with nonpageable virtual storage (ADDRSPC=REAL on the JOB or EXEC statements), the SYSMDUMP output must be directed to a VIO data set.
  - When using IPCS, the data set disposition affects the collection of events. (See "Requesting an Abnormal Termination Dump" in this publication.)
- If you are using the 3800 printer, and want a dump with 204 characters per line, code CHARS=DUMP. If you want a dump with 8 lines per inch, code FCB=STD3. Refer to the sections on the CHARS and FCB parameters for examples of requesting dumps with more data per page.

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 through the standard output class A.

```
//SYSMDUMP  DD      DSNAME=DUMP,DISP=(NEW,KEEP),
//                  UNIT=2400,VOL=SER=147958
```

This step causes the dump to be stored on a standard labeled tape.

```
//STEP1     EXEC    PGM=PROGRAM1
//SYSABEND  DD      DSNAME=DUMP,UNIT=2314,DISP=(,PASS,KEEP),
//                  VOLUME=SER=1234,SPACE=(TRK,(40,20))
//STEP2     EXEC    PGM=PROGRAM2
//SYSABEND  DD      DSNAME=*.STEP1.SYSABEND,DISP=(OLD,DELETE,KEEP)
```

The SYSABEND DD statement specifies that you want the dump stored. The space request in STEP1 is large (40 tracks or 340 maximum tracks) so that the dumping operation will not be inhibited due to insufficient space; if STEP1 does not abnormally terminate but STEP2 does, the dump will be written using the space allocated in STEP1. In both steps, a conditional disposition of KEEP is specified. This will allow storing of the dump if either of the steps abnormally terminates. If both of the steps are successfully executed, the second subparameter of the DISP parameter (DELETE) in STEP2 will delete the data set and free the space acquired for dumping.

```
//STEP1      EXEC     PGM=WWK
//SYSUDUMP   DD       DSNAME=DUMP,UNIT=2314,DISP=( ,DELETE,
//                    KEEP),VOLUME=SER=54366,SPACE=(1680,(160,80))
//STEP2      EXEC     PGM=PRINT,COND=ONLY
//IN         DD       DSNAME=*.STEP1.SYSUDUMP,DISP=(OLD,DELETE),
//                    VOLUME=REF=*.STEP1.SYSUDUMP
```

Step 1 specifies that the dump is to be stored if the step abnormally terminates. Because COND=ONLY is specified in STEP2, the step will be executed only if STEP1 abnormally terminates. STEP2 uses a program that prints the dump.


```
//STEP       EXEC     PGM=EXSYSM
//SYSMDUMP   DD       UNIT=3330,VOLUME=SER=123456,SPACE=(CYL,(0,1)),
//                    DISP=(NEW,DELETE,KEEP),DSN=MDUMP
/*
```

The SYSMDUMP DD statement allocates the dump data set, MDUMP, to a direct access device.

```
//STEP       EXEC     PGM=EXSYSMDP
//SYSMDUMP   DD       DSN=SYS1.SYSMDP00,DISP=SHR
/*
```

The SYSMDUMP is directed to the SYS1.SYSMDP00 data set.

Note: If the data set name is SYS1.SYSMDPxx and DISP=SHR, the first SYSMDUMP produced will be retained on the data set. The first SYSMDUMP taken 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 taken.

```
//STEPA      EXEC    PGM=EXSYSM2,ADDRSPC=REAL
//SYSMDUMP   DD      UNIT=SYSDA,SPACE=(CYL,(0,1)),
//                   DISP=(NEW,PASS)
//STEPB      EXEC    PGM=AMDPRDMP,COND=ONLY
//SYSUT1     DD      DSN=*.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 by specifying a VIO-eligible device group (SYSDA) and by not assigning a data set name (therefore, making the data set temporary and eligible for VIO). In STEPB, the dump output is directed to a class A device for printing.

The SYSCHK Facility

DD Statement

The SYSCHK DD statement defines a checkpoint data set written during the original execution of a processing program.

For detailed information about the checkpoint/restart facilities, see **OS/VS2 MVS Checkpoint/Restart.**

```
//SYSCHK     DD
```

General Rules for Coding

- The SYSCHK DD statement must immediately precede the first EXEC statement of the resubmitted job when restart is to begin at a checkpoint. (If restart is to begin at a step, as indicated by the RD= parameter on the EXEC statement, you must not include a SYSCHK DD statement before the first EXEC statement.)
- The SYSCHK DD statement supports cataloged data sets.
- Include a SYSCHK DD statement among the DD statements for a job whenever a deferred checkpoint restart is to occur, that is whenever a job is resubmitted for restart of execution at a particular checkpoint.
- If you include a JOBLIB DD statement, the SYSCHK DD statement must follow it.
- Code the RESTART parameter on the JOB statement; otherwise the SYSCHK DD statement will be ignored.
- The rules for coding parameters on the SYSCHK DD statement depend on whether or not the checkpoint data set is cataloged. These rules are discussed below under separate headings.

Rules for Coding When the Checkpoint Data Set is Cataloged

If the checkpoint data set is cataloged, you must always code the DSNAME and DISP parameters.

- The DSNAME parameter specifies the name of the checkpoint data set.
- The DISP parameter must specify or imply a status OLD and a disposition of KEEP.
- The UNIT parameter specifies the type and the number of devices assigned to the data set.
- The VOLUME parameter specifies the volume(s) on which the data set resides.
- If the checkpoint entry exists on a tape volume other than the first volume of the checkpoint data set, you must indicate this by coding the volume serial number or volume sequence number in the VOLUME parameter. (The serial number of the volume on which a checkpoint entry was written is contained in the console message printed after the checkpoint entry is written.) If you code the volume serial number, you must also code the UNIT parameter, since the system will not look in the catalog for unit information.

Rules for Coding When the Checkpoint Data Set is Not Cataloged

If the checkpoint data set is not cataloged, you must always code the DSNAME, DISP, VOLUME, and UNIT parameters.

- The DSNAME parameter specifies the name of the checkpoint data set. If the checkpoint data set is partitioned, do not include a member name in the DSNAME parameter.
- The DISP parameter must specify or imply a status of OLD and disposition of KEEP.

- The VOLUME parameter specifies the volume serial number of the volume on which the checkpoint entry resides. (The serial number of the volume on which a checkpoint entry was written is contained in the console message printed after the checkpoint entry is written.)
- The UNIT parameter specifies the device to be allocated to the data set.

Examples of the SYSCHK DD Statement

```
//JOB1      JOB     RESTART=(STEP3,CK3)
//SYSCHK    DD      DSNAME=CHLIB,UNIT=2314,
//                  DISP=OLD,VOLUME=SER=456789
//STEP1     EXEC
```

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=2400,VOLUME=SER=438291
//STEP1     EXEC
```

The checkpoint data set defined on the SYSCHK DD statement is not cataloged. Note that the SYSCHK DD statement follows the JOBLIB DD statement.

The SYSCKEOV Facility

DD Statement

The SYSCKEOV DD statement defines a checkpoint data set for checkpoint records that are generated by the checkpoint at end-of-volume facility. The checkpoint at end-of-volume facility is invoked by the CHKPT parameter.

For detailed information about checkpoint/restart facilities, see **OS/VS2 MVS Checkpoint/Restart.**

```
//SYSCKEOV  DD
```

General Rules for Coding

- The SYSCKEOV DD statement must be specified if one or more other DD statements in a job step are coded with the CHKPT=EOV parameter.
- The SYSCKEOV DD statement must define a sequential BSAM data set.
- The DCB parameter should not be specified. (All DCB information is provided by the checkpoint at end-of-volume facility).
- The LABEL parameter, if coded, must specify SL (IBM standard labels).
- If direct access is specified, it should not be shared with another processor.
- The CHKPT=EOV parameter should not be specified.
- The DISP=MOD parameter is recommended to reduce loss of checkpoint data in the event of a system failure during checkpointing.

Example of the SYSCKEOV DD Statement

```
//SYSCKEOV  DD      DSNAME=CKPTDS,UNIT=TAPE,DISP=MOD
```

Defines a checkpoint data set for checkpoint at end-of-volume records.

The * Parameter—*positional, optional*

The * parameter specifies that data for a processing program follows the DD statement. The * parameter causes the system to check for an input delimiter (/*, //, or when the card reader runs out of cards or whatever you specify on the DLM parameter that overrides /*) on the input reader device.

```
//ddname    DD  *
```


General Rules for Coding

- You can code more than one DD * statement for each job step.
- Code the DATA parameter instead of the * parameter when the data contains statements starting with //.
- When preceding the data with a DD * statement, a delimiter statement (/*) following the data is optional.
- You must code input stream data records in BCD or EBCDIC.
- If the processing program does not read all the data in an input stream, the remaining data is skipped without causing abnormal termination of the job.
- The DLM parameter can be used to define other than the standard delimiter.
- The DSID and the VOL=SER parameters can be used to indicate to a diskette reader that a diskette data set is to be merged into the JCL stream following this DD statement.
- Do not make backward references to a DD * statement using the DCB or Data Set Name parameters.

Rule for Coding a Cataloged or In-Stream Procedure

- A cataloged or in-stream procedure cannot contain a DD * statement. When you call a procedure you can include data in the input stream for each procedure step by adding one or more DD * statement, to each procedure step.

Restriction when Coding *

- The keywords allowed on the DD * statement are: DLM, DCB=BLKSIZE, DCB=BUFNO, DCB=LRECL, VOL=SER, and DSID. All other keywords will cause an error.
- The VOL=SER, DCB=BUFNO, and DSID parameters are ignored except when they are detected by a diskette reader as a request for an associated data set as described in **OS/VS2 IBM 3540 Programmer's Reference.**
- The only parameters you can specify for JES3 SNA RJP input devices are DCB=BLKSIZE and DCB=LRECL. If you do not specify these parameters, default values are provided for JES2 and JES3 devices.

Separating Groups of Data

You can include several distinct groups of data in the input stream for a job step or a procedure step. The system will recognize each group of data if you precede each group with a DD * statement, or if you follow each group with a delimiter statement (/*), or both. (If you leave out the DD * statement for a group of data, the system provides a DD * statement having SYSIN as its ddname.)

*Examples of the * Parameter*

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

Defines several groups of data in the input stream.

```
//STEP2             EXEC    PROC=FRESH
//SETUP.WORK        DD      UNIT=2400,LABEL=( ,NSL)
//SETUP.INPUT1      DD      *
                .
                .
            data
                .
                .
/*
//PRINT.FRM         DD      UNIT=180
//PRINT.INP         DD      *
                .
                .
            data
                .
                .
/*
```

Defines data in the input stream. The input data defined by the DD statement named SETUP.INPUT1 is for use by the cataloged procedure step named SETUP; the input defined by the DD statement named PRINT.INP is for use by the cataloged procedure step named PRINT.

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

Defines data in the input stream.

The AMP Parameter—*keyword, optional*

The AMP parameter completes information in an access method control block (ACB). The ACB is a control block for entry-sequenced, key-sequenced, and relative record data sets.

For further information on AMP and the ACB, see **OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.**


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

AMORG
indicates that the DD statement defines a VSAM data set. AMORG is required to open a DCB (through the ISAM interface program); however, it is not required to open an ACB for a VSAM data set.

BUFND
specifies the number of I/O buffers to be used for transmitting the contents of data control intervals between virtual and auxiliary storage. A minimum of two data buffers is required. If the number of buffers is not specified in the AMP parameter or the ACB or GENCB macro instructions, the default is the number specified for STRNO, plus one additional buffer.

BUFNI
specifies the number of I/O buffers to be used for transmitting the contents of index control intervals between virtual and auxiliary storage. A minimum of one index buffer is required. If the number of index buffers is not specified in the AMP parameter or ACB or GENCB macro instructions, the default is the number specified for STRNO. If the ISAM interface program is being used, a search of the high-level index in virtual storage can be simulated by adding one additional index buffer.

BUFSP
specifies the maximum size of the user area to be allocated for data and index buffers. If you specify less space than was specified in the BUFFERSPACE parameter of the DEFINE command of Access Method Services when the data set was defined, the BUFFERSPACE amount has precedence.

CROPS
specifies one of four checkpoint/restart options, described in **OS/VS2 MVS Checkpoint/Restart.**

RCK
specifies that a data-erase test and data set post-checkpoint modification tests are performed. RCK is the default for CROPS.

NCK
specifies that data set post-checkpoint modification tests are not performed.

NRE
specifies that a data-erase test is not performed.

NRC
specifies that neither a data-erase test nor data set post-checkpoint modification tests are performed.

If no value for CROPS is specified, RCK is assumed. If you specify an option which is not applicable for a data set, such as the data-erase test for an input data set, the option is ignored.

OPTCD
specifies how records flagged for deletion are to be processed with an ISAM processing program using the ISAM interface.

I
specifies, when coded along with OPTCD=L in the DCB, that records marked for deletion by your processing program are not written into the data set by the ISAM interface. If OPTCD=I is specified in the AMP parameter, but OPTCD=L is not specified in the processing program's DCB, records flagged for deletion are treated like any other records; that is, AMP='OPTCD=I', with L not specified, has no effect.

L
specifies that a record marked for deletion by your processing program is to be kept in the data set. Although this parameter has the same meaning and restrictions for ISAM interface as it has for ISAM, it may have to be specified in the AMP parameter when it wasn't previously needed in the ISAM job control language. It is required when OPTCD=L is not specified in the DCB processing program because OPTCD is not merged into the DSCB when ISAM interface is used.

IL
specifies that if the processing program marks a record for deletion, the ISAM interface does not put the record into the data set.

RECFM
specifies the ISAM record format that the processing program is coded for. Although this parameter has the same meaning and restrictions for the ISAM interface as it has for ISAM, it may have to be specified in the AMP parameter when it wasn't previously required in the ISAM job control language. RECFM is required when it is not specified in the DCB in the processing program because RECFM is not merged into the DSCB when the ISAM interface is used. All VSAM requests are for unblocked records. If your program issues a request for blocked records, the ISAM interface sets the overflow-record indicator for each record to indicate that each is being passed to your program unblocked. If RECFM isn't specified in the AMP parameter or in the processing program's DCB, V is the default.

F
indicates fixed-length records.

FB
indicates blocked fixed-length records.

V
indicates variable-length records.

VB
indicates blocked variable-length records.

STRNO
indicates the number of VSAM requests that require concurrent data set positioning. STRNO is an operand of the ACB or GENCB macro instruction and is fully described in **OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.**

SYNAD
is an operand of the EXLST macro instruction. It can be used to override the address of 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 AMP SYNAD parameter is ignored.
You can also use this parameter when processing a VSAM data set with an ISAM processing program to provide an ISAM SYNAD routine or to replace one with another.

TRACE
specifies that the generalized trace facility (GTF) executes with your job to gather information about opening and closing of data sets, and end-of-volume processing. You can print the trace output with the AMDPRDMP program (see **OS/VS2 System Programming Library: Service Aids**).

Rules for Coding

- If the number of buffers specified in the BUFND and BUFNI subparameters causes the virtual storage requirements to exceed the BUFSP specification, the number of buffers is reduced to comply with BUFSP. If BUFSP specifies more space than required by BUFND and BUFNI, the number of buffers is increased.
- 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.
- Apostrophes must enclose each subparameter or group of subparameters if they contain special characters, for example, AMP='BUFSP=value'.
- If the subparameters continue from one line to another, each line of subparameters must begin and end with an apostrophe and the entire group of subparameters must be enclosed in parentheses.

Additional rules for coding and further explanation of the AMP parameter are in the **OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide**.

Examples of the AMP Parameter

```
//AMPDD     DD      DSN=SYS1.MACLIB,DISP=SHR,AMP=( 'BUFSP=200,BUFND=2',
//                  'BUFNI=3,STRNO=4,SYNAD=ERROR' )
```

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 analysis routine. ERROR overrides the error analysis routine specified in the EXLST macro.

```
//AMPDD     DD      DSN=SYS1.MACLIB,DISP=SHR,AMP=( 'BUFSP=23456,BUFND=5',
//                  'BUFNI=10,STRNO=6,SYNAD=ERROR2,CROPS=NCK,TRACE' )
```

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 OPEN is to provide a module trace.

```
//AMPDD     DD      DSN=SYS1.MACLIB,DISP=SHR,AMP=( 'BUFSP=23456',
//                  'BUFND=5','BUFNI=10','STRNO=6','SYNAD=ERROR2',
//                  'CROPS=NCK','TRACE' )
```

Another way of continuing subparameters from one line to another.

The BURST Parameter — *keyword, optional*

The BURST parameter specifies which stacker of the 3800 Printing Subsystem the paper output is to go to. If the stacker specified is different from that which was last requested, or if no previous request has been made, a message is printed on the system console telling the operator to thread the paper into the Burster-Trimmer-Stacker or the continuous forms stacker.

```
BURST= {Y}
       {N}
```

Y
indicates that the printed output is to be burst into separate sheets.

N
indicates that the printed output is to be in continuous, fanfold mode.

Default: If using SYSOUT, default is supplied by the subsystem handling the output. If not using SYSOUT, default is N.

Rules for Coding

- The BURST parameter can be specified on a SYSOUT DD statement when the 3800 is allocated as a system output device or on an output DD statement when the 3800 is allocated as a direct output device.
- The Burster-Trimmer-Stacker can also be requested on the STACKER parameter of the JES3 FORMAT PR statement or the BURST parameter of the JES2 OUTPUT statement.
- Refer to Figure 24 for parameters that are mutually exclusive with BURST.

Example of the BURST Parameter

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

Requests that the paper output be sent to the Burster-Trimmer-Stacker of the 3800 Printing Subsystem. The printed output is separated into individual sheets instead of being stacked in continuous, fanfold mode.

The CHARS Parameter — *keyword, optional*

The character sets in the 3800 Printing Subsystem are used by means of character arrangement tables. The CHARS parameter specifies the name or names of character arrangement tables for printing a data set with the 3800.

For further information on character arrangement tables, see the section, "Writing Output Data Sets" for either JES2 or JES3, and the **IBM 3800 Printing Subsystem Programmer's Guide.** Refer to **OS/VS2 System Programming Library: System Generation Reference** for information on how to choose, at SYSGEN time, the particular groups (other than the Basic group, which is always available).


```
CHARS=(table name [,table name...])
```

table name
name of the character arrangement table (1-4 alphameric or national characters).

General Rules for Coding

- The CHARS parameter can be specified on a SYSOUT DD statement when the 3800 is allocated as a system output device or on an output DD statement when the 3800 is allocated as a direct output device.
- If the output device might not be a 3800 (for example, if printing might be reassigned to a 3211), see the **IBM 3800 Printing Subsystem Programmer's Guide** for restrictions that apply.
- From 1 to 4 table names can be specified on the CHARS parameter.
- Null positions in the CHARS operands are invalid.
- Parentheses are not needed when one table name is coded.
- CHARS can also be coded on the JES3 FORMAT PR statement or the JES2 OUTPUT statement.
- Refer to Figure 24 for parameters that are mutually exclusive with CHARS.

Requesting a High-Density Dump

- In order to request a high-density dump to the 3800, specify CHARS=DUMP on the dump-related DD statement. This will result in 204-character print lines.
- You can also request dump output at 8 lines per inch by coding FCB=STD3. This can be coded on the same DD statement with CHARS=DUMP, or independently.
- DUMP must be the first table specified if more than one table is coded on the CHARS parameter.

Examples of the CHARS Parameter

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

Specifies two character arrangement table names that can be used when printing a data set. GS10 refers to a Gothic character set, GU12 refers to a Gothic-underscored character set.

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

Requests a high-density dump at 8 lines per inch and 204 characters per line.

The CHKPT Parameter—*keyword, optional*

The CHKPT parameter is used to invoke the checkpoint at end-of-volume facility. It specifies that checkpoints are to be taken for the data set defined by the DD statement on which it is coded. For more information, see **OS/VS2 MVS Checkpoint/Restart.**

```
CHKPT=EOV
```

EOV
specifies that checkpoints are to be taken at end of volume for that data set.

Rules for Coding

- The CHKPT parameter is specified only for multi-volume data sets using QSAM or BSAM. (CHKPT is ignored for non-multivolume QSAM or BSAM data sets or for ISAM, BDAM, BPAM, or VSAM data sets.)
- Checkpoints can be taken on either input or output data sets.
- For concatenated BSAM or QSAM data sets, CHKPT must be coded on each DD in the concatenation if checkpoint is desired for each DD.
- If this parameter is specified on one or more DD statements in a job step, a SYSCKEOV DD statement must be provided.
- The RD parameter values NC and RNC on the JOB or EXEC statements override the CHKPT parameter.
- The CHKPT parameter overrides cataloged procedure values or START console values for checkpoints at end of volume.
- Refer to Figure 24 for parameters that are mutually exclusive with CHKPT.

Examples of the CHKPT Parameter

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

INDS is a multivolume QSAM (or BSAM) data set for which a checkpoint is to be taken twice—once after end of volume on TAPE01 and once after end of volume on TAPE02.

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

OUTDS is a multivolume data set being created that will require eight volumes. Seven checkpoints will be taken at the end of volumes one through seven.

The COPIES Parameter — *keyword, optional*

The COPIES parameter specifies the number of copies of the data set to be printed, and if printing is done on a 3800, optionally specifies how the copies are to be grouped.

For further information on the use of the COPIES parameter, see "Obtaining Output" for either JES2 or JES3 in this manual. (page 51 & 79)

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


nnn
specifies the total number of copies of the data set to be printed, subject to an installation limit. nnn is ignored for the 3800 if group values are specified.

group value
describes the grouping of the printed copies for the 3800 printer only. Each group value specifies the number of copies of each page that are to be grouped together. (For example, a group value of 3 causes the first page of a data set to be printed three times before printing is started for the second page.) When group values are coded they override nnn. The total number of copies printed equals the sum total of the group values.

Default for nnn: 1

If the nnn parameter of COPIES is omitted or incorrectly coded, it defaults to 1 and a warning message is issued.

General Rules for Coding

- Except for JES3, nnn can range from 1 to 255. The maximum number of copies for JES3 is limited to 254.
- If the group value subparameter is not specified, the number of copies indicated by nnn is printed. The printed output is in page sequence for each copy.
- Parentheses are not needed when only nnn is coded.
- If you request copies of the entire job (on the JES2 JOBPARM statement) as well as additional copies of the data set (on the DD COPIES parameter), and if the data set is part of the job related output, you may receive a number of copies equal to the product of the two requests.
- Number of copies can also be specified on the JES2 OUTPUT and JES3 FORMAT control statements.
- Refer to Figure 24 for parameters that are mutually exclusive with COPIES. COPIES is also mutually exclusive with the AFF, SEP, SPLIT, and SUBALLOC subparameters. (page 356)

Rules for Coding Group Value

- The group value subparameter is for the 3800 printer only.
- Each group value can range from 1 to 255.
- A maximum of eight group values can be coded on the COPIES parameter. Their sum total must not exceed 255.
- A null group value, or a zero or null position within the sublist of group values, is invalid. (For example, COPIES=(5,) and COPIES=(5,(1,0,4)) are invalid.)

Restriction when Using UNIT Parameter

The COPIES parameter is normally used with the SYSOUT parameter. If, however, COPIES is coded on a DD statement with the UNIT parameter, nnn defaults to 1. If printing is done on a 3800 and group values are specified, the number of copies printed is equal to the first group value.

Examples of the COPIES Parameter

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

Requests 32 copies of the data set defined by the DD statement named RECORD when printing on an impact or 3800 printer.

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

When printing on an impact printer, one copy (the default for nnn) is printed and a warning message is issued. If the 3800 is used, 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.

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

If the output device is 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. If the output device is not a 3800, eight separate copies are printed.

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

The 3800 printer prints two copies of each page, since the UNIT parameter is specified.

The DATA Parameter—*positional, optional*

The DATA parameter specifies that data for a processing program is to follow the DD statement. This data can contain statements with the characters // in columns 1 and 2.

```
//ddname    DD      DATA
```


General Rules for Coding

- You can code more than one DD DATA statement for each job step.
- Code the * parameter instead of the DATA parameter when the data does not contain statements starting with //.
- You must code input stream data records in BCD or EBCDIC.
- If the processing program does not read all the data in an input stream, the remaining data is skipped without causing abnormal termination of the job.
- You must code a delimiter to end data in the input stream. Code /* in columns 1 and 2 or use the DLM parameter.
- The DSID and VOL=SER parameters can be used to indicate to a diskette reader that a diskette data set is to be merged into the JCL stream following this DD statement.
- Do not make backward references to a DD DATA statement using the DCB or Data Set Name parameters.

(see p. 191)

Rules for Coding a Cataloged or In-Stream Procedure

A cataloged or in-stream procedure cannot contain a DD DATA statement. When you call a procedure, you can include data in the input stream for each procedure step by adding one or more DD DATA statements to each procedure step.

Restrictions when Coding DATA

- The keywords allowed on the DD DATA statement are: DLM, DCB=BLKSIZE, DCB=BUFNO, DCB=LRECL, VOL=SER, and DSID and DCB=MODE=C for JES3 only. All other keywords cause an error.
- The VOL=SER, DCB=BLKSIZE, DCB=BUFNO, DCB=LRECL, and DSID parameters are ignored except when they are detected by a diskette reader as a request for an associated data set as described in **OS/VS2 IBM 3540 Programmer's Reference.**

Separating Groups of Data

You can include several distinct groups of data in the input stream for a job step or a procedure step. Precede each group of data with a DD DATA statement and follow it with a delimiter statement (/*). The data contained between the DD DATA statement and the delimiter statement must not contain /* in columns 1 and 2. See the DLM parameter.

Examples of the DD DATA Parameter

```
//INPUT             DD      DATA
                     .
                     .
                   data
                     .
                     .
/*
```

Defines data in the input stream.

```
//STEP2             EXEC    PROC=UPDATE
//PREP.DD4          DD      DSNAME=A.B.C,VOLUME=SER=D88,
//                          UNIT=2314,SPACE=(TRK,(10,5)),
//                          DISP=(,CATLG,DELETE)
//PREP.INPUT        DD      DATA
                .
                .
            data
                .
                .
/*
//ADD.IN            DD      *
                .
                .
            data
                .
                .
/*
```

Defines data in the input stream. The input defined by the DD statement name PREP.INPUT is for use by the cataloged procedure step name PREP. This data contains job control statements. The input defined by the DD statement named ADD.IN is for use by the cataloged procedure step named ADD. Since this data is defined by a DD * statement, it must not contain job control statements.

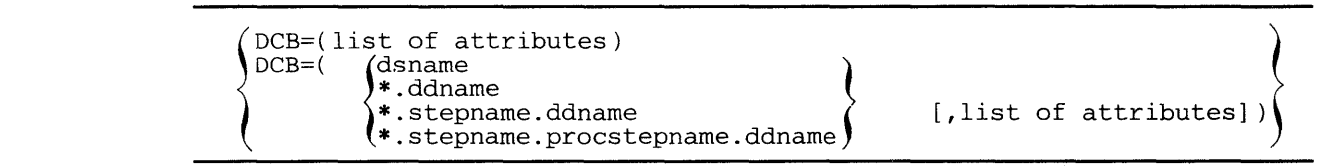
```
//INPUT             DD      DATA
                .
                .
            data
                .
                .
/*
//INPUT3            DD      DATA
                .
                .
            data
                .
                .
/*
```

Defines several groups of data in the input stream.

The DCB Parameter—*keyword, optional*

The DCB parameter is used to complete information about a data set in a data control block (DCB) at execution time. The data control block is originally constructed in a processing program by a DCB macro instruction.

For further information on the formation of the data control block, see **OS/VS2 MVS Data Management Services Guide.**



list of attributes
those DCB keyword subparameters that describe the data set and are needed to complete the data control block. The DCB keyword subparameters are listed alphabetically in this section in the pages immediately following.

dsname
the name of a non GDG cataloged data set from which the system is to copy DCB information. (Generation data group (GDG) names are invalid.) The information is contained in the data set label of the cataloged data set; the data set must reside on a direct access volume and the volume must be mounted before execution of the job step.

***.ddname**
the name of an earlier DD statement in the same job step from which the system is to copy DCB information. The earlier DD statement must not have the name of a cataloged data set as a DCB subparameter.

***.stepname.ddname**
the name of a DD statement (ddname) in a earlier job step (stepname) from which the system is to copy DCB information. The earlier DD statement must not have the name of a cataloged data set as a DCB subparameter.

***.stepname.procstepname.ddname**
the name of a DD statement (ddname), which appears in a procedure step (procstepname); the procedure step is part of a cataloged procedure that was called by an earlier job step (stepname). The earlier DD statement must not have the name of a cataloged data set as a DCB subparameter.

General Rule for Coding

Parentheses are not needed if only one keyword subparameter, a data set name, or a backward reference is coded.

Completing the Data Control Block

- You must code the DCB macro instruction in a processing program written in assembler language. However, some DCB operands (particularly those that may change from one execution of the program to the next) can be specified as DCB subparameters on a DD statement, or read from a data set label, or both.
- If your processing program is written in a programming language other than assembler, DCB operands might be specified as part of file definition statements in your program, as DCB subparameters on a DD statement or data set label, or might be taken from language-defined default values via the DCB open exit routine. (Refer to the Programmer's

Guide for your language to learn the alternatives open to you. Refer to **OS/VS2 MVS Data Management Services Guide** for a description of the DCB open exit routine.)

- You must code the DCB parameter on the DD statement if the data control block is not completed by your processing program, the data set label, or your languages's defined values. There are several ways of specifying DCB information on the DD statement. The following methods are explained in detail in the next three groups of syntax rules:
  - Supplying all pertinent DCB keyword subparameters on the DD statement.
  - Copying the DCB information from the data set label of an existing cataloged data set.
  - Copying the DCB information from an earlier DD statement.

Supplying DCB Keyword Subparameters

- Code the information required to complete the data control block as keyword subparameters in the DCB parameter.
- Separate DCB keyword subparameters by a comma.
- If the processing program and the DCB parameter supply the same subparameter, the subparameter on the DD statement is ignored. If the DCB parameter and the data set label supply the same subparameter, the subparameter on the data set label is ignored.
- All DCB subparameters, except BLKSIZE, BUFNO, and DIAGNS are mutually exclusive with the DDNAME parameter.
- The DCB keyword subparameters are listed alphabetically in this section in the pages immediately following.

Copying DCB Information From the Label of a Cataloged Data Set

- You can copy DCB information from the data set label of a cataloged data set on a currently mounted direct access volume. A permanently resident volume is the most likely place from which to copy information because it is always mounted.
- Code in the DCB parameter the data set name of the cataloged data set. You may not specify a generation data group name. The data set name cannot contain special characters, except for periods used in a qualified name.
- The following DCB keyword subparameters can be copied from the data set label:

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

The volume sequence number, system code, and expiration date of the cataloged data set will also be copied unless you specify them in the DD statement.

- If you code any DCB keyword subparameters following the name of the cataloged data set, these subparameters override any of the corresponding subparameters that were copied.
- The DCB subparameters are listed alphabetically in this section in the pages immediately following.

Copying DCB Information From an Earlier DD Statement

- The earlier DD statement from which DCB information can be copied can be contained in the same job step, an earlier job step, or an earlier cataloged procedure step. Code in the DCB parameter one of the following types of reference names, depending on the location of the DD statement you want to use:

  *.ddname
  *.stepname.ddname
  *.stepname.procstepname.ddname

- If you code any DCB keyword subparameters following the reference to the DD statement, these subparameters override any of the corresponding subparameters that were copied.
- The system copies only those subparameters from the earlier DD statement that are not again specified on the referencing DD statement.
- The UCS and FCB parameters are also copied from the earlier DD statement unless they are overridden by the referencing DD statement.
- Do not make backward references to a DD * or a DD DATA statement.
- The DCB subparameters are listed alphabetically in this section in the pages immediately following.

Examples of the DCB Parameter


```
//DD1    DD       DSNAME=ALP,DISP=( ,KEEP),VOLUME=SER=44321,
//                UNIT=2400,DCB=(RECFM=FB,LRECL=240,BLKSIZE=960,
//                DEN=1,TRTCH=C)
```

Defines a new data set and contains the information necessary to complete the data control block.

```
//DD2    DD       DSNAME=BAL,DISP=OLD,DCB=(RECFM=F,LRECL=80,
//                BLKSIZE=80)
//DD3    DD       DSNAME=CNANN,DISP=( ,CATLG,DELETE),UNIT=2400,
//                LABEL=( ,NL),VOLUME=SER=663488,DCB=*.DD2
```

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

```
//DD4    DD       DSNAME=JST,DISP=(NEW,KEEP),UNIT=2314,
//                SPACE=(CYL,(12,2)),DCB=(A.B.C,KEYLEN=8)
```

Defines a new data set and requests the system to copy 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 since KEYLEN is coded on the DD statement.

```
//DD5    DD       DSNAME=SMAE,DISP=OLD,UNIT=2314,
//                DCB=(*.STEP1.PROCSTP5.DD8,BUFNO=5)
```

Defines an existing data set and requests the system to copy the DCB subparameters from the DD statement named DD8, which is contained in the procedure step named PROCSTP5. The cataloged procedure was called by the job step named STEP1. If any of the DCB subparameters coded on the procedure DD statement have been previously defined for this data set, they are ignored. If the BUFNO subparameter has not been previously specified for the data set, then five buffers are assigned to the data control block.

The following is a brief description of the DCB subparameters. For more detail on each of them, refer to the book describing the access method you are using.

Access Method	Publication
BDAM, BISAM, BPAM, BSAM, QISAM, QSAM	**OS/VS2 MVS Data Management Macro Instructions**
TCAM	**OS/VS2 TCAM Programmer's Guide**
GAM	**Graphic Programming Services for 2250** **Graphic Programming Services for 2260**
EXCP	**OS/VS2 System Programming Library: Data Management**
BTAM	**OS/VS BTAM**

Sub-parameters \ 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 supplied from the same source. Default: D (doubleword)
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. It can only be coded when RECFM=VS. If both BFALN and BFTEK are specified, they must be supplied from the same source.
BLKSIZE	X		X	X		X		X	X	X	BLKSIZE=number of bytes Specifies the maximum length, in bytes, of a block. The minimum length is 18. The largest number allowed is 32,760 except for blocks of ASCII records on magnetic tape. This maximum length is 2048. 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 subparameter overrides the block size specified in the label. BLKSIZE may be coded but will have no effect on EXCP processing.
BUFIN										X	BUFIN=number of buffers Specifies the number of buffers to be assigned initially for receiving operations for each line in the line group. The number of buffers specified in the combined BUFIN and BUFOUT operands must be no greater than the number of buffers in the buffer pool for this line group (not including those for disk activity only). Default: 1
BUFL	X	X	X	X		X		X	X	X	BUFL=number of bytes Specifies the length, in bytes, of each buffer in the buffer pool. The maximum length allowed is 32,760.
BUFMAX										X	BUFMAX=number of buffers Specifies the maximum number of buffers to be allocated to a line at one time. The number must be greater than 1 but may not exceed 15. It must be at least equal to the larger of the numbers specified by the BUFIN and BUFOUT subparameters. Default: 2
BUFNO	X	X	X	X	X	X		X	X		BUFNO=number of buffers Specifies how many buffers are to be assigned to the DCB; the maximum normally is 255, but can be less because of the size of the partition or region.
BUFOFF				X					X		BUFOFF=(n|L) Specifies the buffer offset; that is, the length of an optional block prefix that can precede a block of one or more ASCII records on magnetic tape. For input, n can be 0 through 99, unsigned. For output, n can only be 0. L can be specified only when RECFM=D, indicating a four byte field containing block length.
BUFOUT										X	BUFOUT=number of 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
BUFSIZE										X	BUFSIZE=number of bytes Specifies the length, in bytes, of each of the buffers to be used for all lines in a particular line group. This length must be at least 31 bytes, but cannot exceed 65,535.

Sub-parameters \ Access Method	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
CODE				X		X			X		CODE={A|B|C|F|I|N|T} Specifies the paper tape code used for punched data. The subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive. A ASCII (8 track) B Burroughs (7 track) C National Cash Register (8 track) F Friden (8 track) I IBM BCD perforated tape transmission code (8 track) N No conversion required T Teletype[1] (5 track) Default: I
CPRI										X	CPRI={R|E|S} Specifies the relative transmission priority assigned to the lines in this line group. R Specifies that CPU receiving has priority over CPU sending. E Specifies that receiving and sending have equal priority. S Specifies that CPU sending has priority over CPU receiving. *Note:* Subparameter CPRI is mutually exclusive with subparameter THRESH and with DD parameter OUTLIM.
CYLOFL								X			CYLOFL=number of tracks Specifies how many tracks on each cylinder are to hold the records that overflow from other tracks on that cylinder. The maximum is 99. Specify CYLOFL only when OPTCD=Y.
DEN				X		X			X		DEN={0|1|2|3|4} Specifies the magnetic density in number of bits-per-inch used to write a data set. DEN: 0 — 7-track tape: 200 — 9-track tape: — 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 is for non-return-to-zero inverted recording mode. PE is for phase encoded recording mode. GCR is for 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.
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. When GTF is not running and tracing user events, DIAGNS is ignored.

[1] Trademark of Teletype Corporation, Skokie, Ill.

Sub-parameters \ Access Method	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
DSORG	X	X	X	X	X	X	X	X	X	X	DSORG=data set 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. 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
EROPT					X				X		EROPT=n BTAM: Requests the BTAM on-line terminal test option. n=T QSAM: Specifies the option to be executed if an error occurs in reading or writing a record. n=ACC system is to accept the block causing the error. SKP system is to skip the block causing the error. ABE system is to cause abnormal end of task. Default: ABE
FRID				X							FRID=identifier Specifies a 1 to 4 character load module name identifying the first format record of the 3886 data set. FRID is mutually exclusive with the FCB parameter.
FUNC				X					X		FUNC={I|R|P|W|D|X|T} Specifies the type of data set to be opened for the 3305/3525 card reader/card punch. Unpredictable results will occur if coded with other than the 3505/3525 devices. *Note:* Subparameter FUNC is mutually exclusive with the data set sequence number of the LABEL parameter. I data in a data set is to be punched into and printed on cards. R data set is for reading cards. P data set is for punching cards. W data set is for printing. D data protection for a punch data set. X data set is for both punching and printing. T two-line print option. Default: output data set is P; input data set is R. 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
GNCP							X				GNCP=number of channel programs Specifies the maximum number of input/output macro instructions that will be issued before a WAIT macro instruction.
INTVL										X	INTVL={integer|0} Specifies the number of seconds of delay between passes through an invitation list. Default: 0

Sub-parameters \ Access Method	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
KEYLEN	X		X	X		X		X		X	KEYLEN=number of bytes Specifies the length, in bytes, of the keys used in a data set. The largest number allowed is 255. The key length information can be supplied from the data set label for an existing data set. If a key length is not specified, no input or output requests that require a key can be used. The subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive.
LIMCT	X										LIMCT=number of blocks or 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 the extended search option is specified (OPTCD=E); otherwise, LIMCT is ignored. If the number specified in the LIMCT subparameter equals or exceeds the number of blocks or tracks in the data set, the entire data set is searched.
LRECL			X	X		X		X	X	X	LRECL=number of bytes Specifies the length, in bytes, for fixed-length records or it specifies the maximum length, in bytes, for variable-length records. The length cannot exceed the blocksize (BLKSIZE) for U- or F- format records. It cannot exceed BLKSIZE-4 for V- or D- format records. For VS- format records, LRECL may exceed BLKSIZE. For unblocked records with a relative key position (RKP) of zero, the record length includes only the data portion of the record. QSAM: LRECL=X Specifies that the logical record length exceeds 32,760 bytes for variable-length spanned records.
MODE				X		X			X		MODE= {C [O]} {E [R]} Specifies the mode of operation to be used with a card reader, a card punch, or a card read-punch. C indicates card image (column binary) mode. E indicates EBCDIC mode. O indicates optical mark read mode. R indicates read column eliminate mode. If R is specified then either C or E must be specified. Do not code the MODE subparameter for data entered through the input stream except when using JES3. The subparameters MODE, CODE, KEYLEN, PRTSP, STACK, and TRTCH are mutually exclusive. Do not code MODE=C for JES2 output. Default: E
NCP		X	X	X							NCP=number of channel programs Specifies the maximum number of READ or WRITE macro instructions that can be issued before a CHECK macro instruction is issued to test for completion of the I/O operation. The largest number that can be specified is 99, but may be less depending on the size of the region or partition. If chained scheduling is used, NCP must be greater than 1. Default: 1
NTM								X			NTM=number of 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. This information is required only when the master index option (OPTCD=M) has been selected. If you omit NTM information and OPTCD=M is specified, the master index option is ignored.

Sub-parameters \ Access Method	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
OPTCD	X	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 by one method, that is, by the data set label of an existing data set, the DCB macro, or the DD DCB parameter. However, it can be modified by the problem program. The characters may be coded in any order and when used in combination, no commas are permitted between characters. BDAM: OPTCD=$\left\{\begin{Bmatrix}A\\R\end{Bmatrix}\ [E]\ [F]\ [W]\right\}$ 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 or available space. (LIMCT must be coded but do not code LIMCT=0 because it will cause an ABEND when a READ or WRITE macro instruction is issued.) F indicates that feedback can be requested in READ and WRITE macro instructions and the device address returned is to be in the same form as that presented to the control program. W requests a validity check for write operations on direct access devices. BISAM: OPTCD={[L] [R] [W]} L requests that the control program 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, 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 data control block. (The problem program can analyze these statistics to determine when to reorganize the data set. This option is provided whenever the OPTCD subparameter is omitted from all sources.) W requests a validity check for write operations on direct access devices. BPAM: OPTCD={C|W|CW} {C|H|HC} {C|WH|WHC} C requests that chained scheduling be used W requests a validity check for write operations on direct access devices. H requests that a partitioned data set which is being processed and resides on MSS, if opening for input, is to be staged to EOF on the virtual DASD. Otherwise only the directory is staged. 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 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 that chained scheduling be used. H requests hopper empty exit for Optical Readers or bypass of DOS checkpoint records.

Sub-parameters \ Access Method	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
OPTCD (continued)											J for use with the 3800 Printing Subsystem — instructs the system that the first byte of each output data line (following the print control character) is a table reference character, which selects the character arrangement table corresponding to the order in which the table names have been specified on the CHARS parameter. For considerations before using OPTCD=J, see the ***IBM 3800 Printing Subsystem Programmer's Guide.*** Q requests that ASCII tape records in an input data set be converted to EBCDIC code when the input record has been read, or 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 feature and the 3800; unblocks data checks and allows analysis by an appropriate error analysis routine. (If U is omitted, data checks are blocked, that is, not recognized as errors.) for MSS — requests window processing to reduce the amount of staging space required to process large sequential data sets on MSS. DSORG must be PS, allocation must be in cylinders, and type of I/O accessing must be INPUT only or OUTPUT only. W requests a validity check for write operations on direct access devices. Z for magnetic tape input — requests the control program to shorten its normal error recovery procedure. When specified, a data check is considered permanent after five unsuccessful attempts to read a record. for direct access storage device input — specifies search direct (SD) for sequential data sets. for direct access input — requests search direct technique. OPTCD=Z is ignored if chained scheduling is used. BTAM and EXCP: OPTCD=Z Z for magnetic tape input — requests the control program to shorten its normal error recovery procedure. When specified, a data check is considered permanent after five unsuccessful attempts to read a record. BTAM Only: for direct access storage device input — specifies search direct (SD) for sequential data sets. 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, RKP must be greater than zero for fixed-length records and greater than four for variable-length records.)

Sub-parameters \ Access Method	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
OPTCD (continued)											M requests that the system create and maintain a master index(es) according to the number of tracks specified in the NTM subparameter. R requests that the control program place reorganization criteria information in certain fields of the data control block. (The problem program can analyze these statistics to determine when to reorganize the data set. This option is provided whenever the OPTCD subparameter is omitted from all sources.) U requests that the system accumulate track index entries in storage and write them as a group for each track of the track index. This can only be specified 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. TCAM: OPTCD={C|U|W} C specifies that one byte of the work area be used to indicate if a segment of a message is the first, middle, or last segment. U specifies that the work unit to be handled 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 eight-byte field in the work area.
PCI										X	PCI={((N|R|A|X)[,N|,R|,A|,X])} Specifies whether or not a program-controlled interruption (PCI) is to be used to control the allocation and freeing of buffers; the PCI subparameter also specifies how these operations are to be performed. The operands shown in the format above apply to receiving and sending operations, respectively. N specifies that no PCIs are taken during filling (on receiving operations) or emptying (on sending operations) of buffers. 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. Parentheses are not needed if only the first operand is coded. Default: (A, A)

Sub-parameters \ 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 that the data control block is used for an online printer. The PRTSP subparameter specifies the line spacing on a printer as 0, 1, 2, or 3. It is valid only if the control characters A and M are not specified in the RECFM subparameter. PRTSP=2 is ignored when specified with SYSOUT. The subparameters PRTSP, CODE, KEYLEN, MODE, STACK, and TRTCH are mutually exclusive. 0 specifies that spacing is suppressed. 1 specifies single spacing. 2 specifies double spacing. 3 specifies triple spacing. Default: 1
RECFM	X		X	X		X		X	X	X	Specifies the format and characteristics of the records in the data set. The format and characteristics must be completely described by one source; that is, the data set label of an existing data set, the DCB macro, or the DD DCB parameter. However, it can be modified by the problem program. BDAM: RECFM={U | V[S] | [BS] | F[T]} U indicates that the records are of undefined length. V indicates that the records are of variable length. VS indicates that the records are of variable length, and spanned. VBS indicates that the records are of variable length, blocked and spanned and the problem program must block and segment the records. F indicates that the records are of fixed length. T indicates that the records may be written using the track-overflow feature. Default: undefined-length, unblocked records. BPAM: RECFM={U[T][A|M] | V[B|T|BT][A|M] | F[B|T|BT][A|M]} A indicates that the record contains American National Standards Institute control characters. B indicates that the records are blocked. F indicates that the records are of fixed length. M indicates that the records contain machine code control characters. T indicates that the records may be written using the track-overflow feature. Chained scheduling (OPTCD=C) will be ignored. U indicates that the records are of undefined length. V indicates that the records are of variable length. Default: U

Sub-parameters \ Access Method	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
RECFM (continued)											BSAM, EXCP, and QSAM: RECFM= { U [T] [A] [M] / F [B] [S] [T] [BS] [BT] [BST] [A] [M] / V [B] [S] [T] [BS] [BT] [BST] [A] [M] } For BSAM, EXCP, and QSAM using ASCII data sets on tape: RECFM= { D [B] [A] / U [A] / F [B] [A] } A or M cannot be specified if the PRTSP subparameter is specified. A indicates that the record contains ANSI device control characters. B indicates that the records are blocked. D indicates that the records are variable-length ASCII tape records. F indicates that the records are of fixed length. M indicates that the records contain machine code control characters. S for fixed-length records, 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. For variable-length records, 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) will be ignored. U indicates that the records are of undefined length. V indicates that the records are of variable length. The V subparameter cannot be used for variable-length ASCII tape data sets (see D above) or for a card reader data set. RECFM=V cannot be used for a 7-track tape unless the data conversion feature (TRTCH=C) is used. Default: U
											QISAM: RECFM= { V [B] / F [B] } B indicates that the records are blocked. F indicates that the records are of fixed length. V indicates that the records are of variable length; variable length records cannot be in ASCII. When indexed sequential data sets are created, you can code the RECFM subparameter; when existing indexed sequential data sets are processed, RECFM must be omitted. Default: V
											TCAM: RECFM= { U / V [B] / F } B indicates that the records are blocked. F indicates that the records are of fixed length. U indicates that the records are of undefined length. V indicates that the records are of variable length. Default: U
RESERVE										X	RESERVE=(number1,number2) Specifies the number of bytes (from 0 to 255) to be reserved in a buffer for insertion of data by the DATETIME and SEQUENCE macros. number1 indicates the number of bytes to be reserved in the first buffer that receives an incoming message. number2 indicates the number of bytes to be reserved in all the buffers following the first buffer in a multiple-buffer header situation. Default: (0, 0)

Sub-parameters \ Access Method	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
RKP						X		X			RKP=number Specifies the position of the first byte of the record key within each logical record. The beginning byte of a record is addressed as 0. If RKP=0 is specified for blocked fixed-length records, the key begins in the first byte of each record, and the delete option (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; the delete option can be specified. For variable-length records, the relative key position must be 4 or greater, when the delete option (OPTCD=L) is not specffied. The relative key position must be 5 or greater if the delete option is specified. Default: 0 RKP can be coded but will have no effect on EXCP processing.
STACK				X		X			X		STACK={1|2} Specifies which stacker bin is to receive a card. The subparameters STACK, CODE, KEYLEN, MODE, PRTSP, TRTCH are mutually exclusive. Default: 1
THRESH										X	THRESH=number Specifies the percentage of the non-reusable disk message queue records to be used before a flush closedown occurs. Default: closedown occurs when 95% of the records have been used. *Note:* Subparameter THRESH is mutually exclusive with subparameter CPRI and with DD parameter OUTLIM.
TRTCH				X		X			X		TRTCH={C|E|T|ET} Specifies the recording technique for seven-track tape. The subparameters TRTCH, CODE, KEYLEN, MODE, PRTSP, and STACK are mutually exclusive. C specifies that the data conversion feature is to be used, with odd parity and no translation. E specifies even parity, with no translation and no conversion. T specifies that BCD to EBCDIC translation is required when reading, EBCDIC to BCD translation when writing; with odd parity and no data-conversion feature. ET specifies even parity and no conversion with BCD to EBCDIC or EBCDIC to BCD translation required. Default: odd parity with no translation or conversion.

The DDNAME Parameter—*keyword, optional*

The DDNAME parameter allows you to postpone the definition of a data set until later in the same job step. In the case of cataloged and in-stream procedures, this parameter allows you to postpone defining a data set in the procedure until the procedure is called by a job step.

```
DDNAME=ddname
```

ddname
the name of the DD statement on which the data set is defined.

General Rules for Coding

- Only the DCB subparameters DIAGNS, BLKSIZE, and BUFNO can be coded with the DDNAME parameter. If this subparameter is coded both on the DD statement that contains the DDNAME parameter and on the DD statement that actually defines the data set, the subparameter coded with the DDNAME parameter is ignored.
- You can code the DDNAME parameter up to five times in a job step or procedure step. However, each time the DDNAME parameter is coded, it must refer to a different ddname.
- If the data set, which will be defined later in the job step, is to be concatenated with other data sets, the DD statements that define these other data sets must immediately follow the DD statement that includes the DDNAME parameter.
- The DDNAME parameter cannot appear on a DD statement named JOBLIB, JOBCAT, or STEPCAT.
- The DDNAME parameter cannot refer to a DD statement that has DYNAM coded on it.
- If you postpone the definition of a data set by coding the DDNAME parameter and then do not define the ddname later in the job step, the DD statement is treated as though you had specified the DUMMY parameter and a warning message is issued. This is also true if the DD statement referred to by the DDNAME parameter appears in the job stream ahead of the statement containing the parameter.
- Refer to Figure 24 for parameters that are mutually exclusive with DDNAME.

Rules for Coding Backward References

- In any backward reference to a data set, you must use the ddname of the DD statement containing the DDNAME parameter, **not** the ddname specified in the DDNAME parameter. The DD statement that actually defines the data set cannot contain any backward references to a DD statement that follows the one with the DDNAME parameter.

For example:

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

DD3 specifically refers to DD1 in its UNIT= and VOL=SER= references. DD3 uses the ddname of the statement containing the DDNAME parameter, not the ddname specified in the DDNAME parameter.

INPUT specifically identifies the volume 333333 in its VOL=SER=333333 reference. It is invalid to refer to volume 333333 as VOL=SER=*.SHOW.DD5 because DD5 follows the DD statement that contains the DDNAME parameter, DD1.

Examples of the DDNAME Parameter

The following procedure step is included in a cataloged procedure named CROWE:

```
//PROCSTEP      EXEC    PGM=FLORA
//DD1           DD      DDNAME=DAVE
//POD           DD      DSNAME=JEREMY,DISP=OLD
```

The DD statement named DD1 is meant to contain weekly records, in the form of data in the input stream, that are processed by this step. Since the * and DATA parameters cannot be included in cataloged procedures, the DDNAME parameter is used to postpone defining the data set until the procedure is called by a job step. When calling the procedure, you would code:

```
//STEPA       EXEC    CROWE
//DAVE        DD      *
              .
              .
              data
              .
              .
/*
```

When there are multiple steps in a procedure, use the form stepname.ddname to define the ddname. For example, the following procedure steps are included in a cataloged procedure named TEA:

```
//STEP1       EXEC    PGM=SUGAR
//DD1         DD      DDNAME=CHUCK
              .
              .
              .
//STEP2       EXEC    PGM=LEMON
//DD2         DD      DSNAME=BETA,DISP=OLD
              .
              .
              .
```

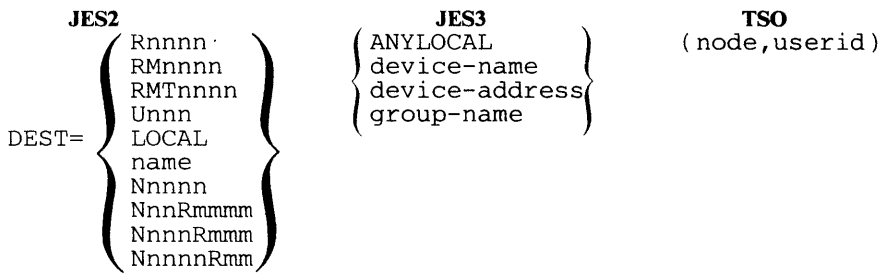
When calling the procedure, you would code:

```
//STEPA           EXEC    TEA
//STEP1.CHUCK     DD  *
                    .
                    .
                    data
                    .
                    .
/*
```

The DEST Parameter—*keyword, optional*

Use the DEST parameter to route output to specified destinations. Using the DEST parameter, you can specify a remote destination (work station), a central computing center destination, a TSO user's location (node and userid), a VM user's location (node and userid), or a specific local device for an output data set.

For further information on the DEST parameter, see "Obtaining Output" for either JES2 or JES3.



JES2

Nnnnn
where nnnn is 1-4 digit decimal number (1-1000) indicating the node to which the output data set is to be directed.

NnnRmmmm, NnnnRmmm, or NnnnnRmm
where N designates a node and R designates a remote work station that is connected to the node.

nn, nnn, or nnnn is 1-99, 1-999, or 1-1000 and mm, mmm, or mmmm is 1-99, 1-999, or 1-1000. The maximum number of digits (n and m combined) can not exceed six. A specification of R0 is equivalent to the specification LOCAL at node N.

Rnnnn, RMnnnn, or RMTnnnn
where nnnn is a 1-4 digit decimal number (1 to 1000) indicating the remote terminal to which the output data set is to be directed.

Note: R0 is equivalent to LOCAL.

Unnn
where nnn is a 1-3 digit decimal number (1-255) indicating the local device with special routing to which the output data set is to be directed.

LOCAL
a local device is the destination for the output data set.

name
1-8 alphameric or national character name of a remote or local device (as defined by the system programmer) to receive the output data set.

JES3

ANYLOCAL
any device (either a printer or punch as defined by the output class on the DD statement) attached to the central CPU to received the output data set.

device-name
1-8 alphameric or national character name of a local printer or punch (as defined by the system programming staff) to receive the output data set.

device-address
three character physical device address of the device to receive the output data set.

group-name
name of a group of local devices, an individual remote station, or a group of remote stations to receive the output data sets. Specify LOCAL to define the default group-name for local devices (that is, those local devices that are in no other group).

Default: name of the source of the job (whoever submitted the request).

If the destination specified is invalid, the job fails.

TSO

node
a valid name consisting of from 1 to 8 characters to which the system directs output.

userid for TSO
a valid userid consisting of from 1 to 7 alphameric or national characters. You must begin the userid with an alphabetic or national character.

userid for VM
a valid userid consisting of from 1 to 8 alphameric, national, or special characters. You can begin userid with a nonalphabetic character.

Rules for Coding

- The DEST parameter can only be coded on a DD statement that includes the SYSOUT parameter. Otherwise, DEST is syntax-checked and ignored.
- Output destination can also be coded on the JES2 OUTPUT and ROUTE control statements and the JES3 MAIN ORG and FORMAT PR and PU DEST parameters.
- You cannot code DEST=(node,userid) on JES2 or JES3 control statements.
- DEST=(node,userid) is invalid with a "program-name" specification on the SYSOUT parameter. For example, you cannot code:

```
//DD1A          DD      SYSOUT=(A,program-name),DEST=(node,userid)
```

- Code the SYSOUT parameter and the node subparameter of the DEST parameter when you code the userid subparameter of the DEST parameter. For example, you must code:

```
//DD2A          DD      SYSOUT=D,DEST=(node,userid)
```

  You cannot code:

```
//DD3A          DD      DEST=(,userid)
//DD4A          DD      SYSOUT=M,DEST=(,userid)
```

- Enclose the userid subparameter in apostrophes when the userid contains special characters. For example,

```
//DD5A          DD      SYSOUT=C,DEST=(PUB,'VM/370')
```

  The DEST parameter directs the output to node PUB and userid VM/370. The userid is enclosed in quotes because the / is a special character.

Example of the DEST Parameter

```
//JOB01         JOB     ,'REBECCA BARNHARDT',MSGLEVEL=1
//STEP1         EXEC    PGM=INTEREST
//DEB           DD      SYSOUT=A
//GWB           DD      SYSOUT=A,DEST=RMT1
```

The workstation from which the job was submitted receives the output described by the DEB DD statement. The user identified by the station-id RMT1 receives the output described by the GWB DD statement.

The DISP Parameter—*keyword, optional*

The DISP parameter describes the status of a data set to the system. It also indicates what is to be done with the data set after termination of the job step or job that processes it. You can indicate in the DISP parameter one disposition to apply if the step terminates normally after execution and another to apply if the step terminates abnormally (conditional disposition). For further information on the DISP parameter, see "Disposition Processing".

```
DISP=   ( [NEW   ] [,DELETE   ] [,DELETE   ] )
          [OLD   ]  [,KEEP   ]  [,KEEP   ]
          [SHR   ]  [,PASS   ]  [,CATLG   ]
          [MOD   ]  [,CATLG  ]  [,UNCATLG ]
          [,     ]  [,UNCATLG]
                    [,     ]
```


Status

Note: The disposition of a data set is solely a function of the DISP parameter; however, the disposition of the volumes on which the data set resides is a function of the volume status when the volume is demounted.

NEW
specifies that the data set is to be created in this job step.

OLD
specifies that the data set existed before this job step and that this step requires exclusive (non-shared) use of the data set.

SHR
specifies that the data set existed before this job step and can be used simultaneously (shared) by another job, since it will only be read.

MOD
specifies an addition to the end of an existing sequential data set or the creation of a new data set. Specifying MOD for an existing data set causes the read/write mechanism to be positioned after the last sequential record. The system creates a new data set if it cannot find volume information on the DD statement, in the system catalog, or passed from a previous step (see "Rules for Coding").

,
specifies that NEW is assumed and that a normal disposition follows.

Normal Termination Disposition

DELETE
specifies that the data set is no longer needed and its space on the volume is to be released at the end of this job step for use by other data sets if the data set has expired (see EXPDT or RETPD parameters).

KEEP
specifies that the data set is to be kept on the volume at the end of this job step.

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

CATLG
specifies that the data set is to be kept at the end of this job step and an entry pointing to the data set is to be placed in the system or user catalog. For CVOL catalogs, any missing index levels will be created. For information about the rules for cataloged data set names, refer to **Planning for Enhanced VSAM** and **OS/VS2 Access Method Services.**

UNCATLG
specifies that the data set is to be kept at the end of this job step but the entry pointing to the data set in the system or user catalog, and unneeded indexes, with the exception of the highest level, are to be deleted.

,
specifies no explicit disposition for the data set, but indicates that a conditional disposition follows. A new data set is deleted and a data set that existed before execution of the job step is kept at step termination.

Abnormal Termination (Conditional) Disposition

DELETE
specifies that the data set is no longer needed and its space on the volume is to be released for use by other data sets if this step abnormally terminates.

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

CATLG
specifies that an entry pointing to the data set is to be placed in the system or user catalog if this step abnormally terminates. For CVOL catalogs, any missing index levels will be created.

UNCATLG
specifies that the entry pointing to the data set in the system or user catalog, and unneeded indexes, with the exception of the highest level, are to be deleted if this step abnormally terminates.

Rules for Coding

- Parentheses are not needed when only the first subparameter is specified.
- If the data set is new, you can omit the subparameter NEW. However, if you specify a disposition or conditional disposition, you must code a comma to indicate the absence of NEW.
- If you specify a disposition of MOD, the system creates a data set when:
  - Volume information is not specified (either by volume serial number or volume reference) and the data set is not cataloged or passed from another job step.
  - A volume reference within the same step refers to a non-specific request and the dsnames in the referencing and referenced statements are not the same. The MOD disposition is specified in the referencing DD statement.
  - A volume reference within the same step refers to a non-specific request for different areas of the same ISAM data set. The MOD disposition is specified in the referencing DD statement.
- The system treats a request specifying MOD as a nonspecific volume request for a new data set if it cannot locate volume information. If, after it chooses a volume, the system finds another data set with the same name on that volume, the system will try to allocate a different volume.
  If the system creates a new data set, the read/write mechanism is positioned at the beginning of the data set the first time it is opened for output. For subsequent OPENs within the same step, the read/write mechanism is positional after the last sequential record.
- You can omit the DISP parameter if a data set is created and deleted during a job step.
- If you do not want to change the automatic disposition processing performed by the system, you need not code the second subparameter. (When the second subparameter is not coded, the system automatically deletes data sets that did not exist before the job.) If you omit the second subparameter and code a conditional disposition, you must code a comma to indicate the absence of the second subparameter.

- You must specify a disposition of PASS or DELETE for a temporary data set or a data set with a system-generated name; that is, when DSNAME=dsname or DSN=dsname is omitted from the DD statement. Any other disposition will be overridden by the system with PASS.
- If a job step abnormally terminates and a conditional disposition is not specified, the normal disposition (second subparameter) is processed.
- If a temporary data set name is specified, any conditional disposition other than DELETE or PASS is ignored.
- A data set can only be passed within a job. VSAM data sets cannot be passed.
- If a job step abnormally terminates during the creation of a data set with a conditional disposition of DELETE and the data set has an unexpired retention period or expiration date, the data set will be deleted.
- If a job step abnormally terminates, conditional dispositions of CATLG, UNCATLG, or DELETE (of a cataloged data set) for passed data sets will not update a user catalog.
- An unopened tape data set can be cataloged unless the request is non-specific, or the data set is allocated to a dual-density tape drive and no density is specified.
- SHR can also be coded as SHARE.
- If MOD is specified and the volume information exists for a multivolume data set, the first volume(s) will be mounted on the unit(s) allocated. However, if the data set is opened for output with a status of MOD, OPEN starts with the last volume. If more volume serial numbers were specified than units allocated, the operator must demount the first volumes. To avoid this inefficiency, specify DEFER in the UNIT parameter, VOL=REF (for tape data sets only), or an explicit volume sequence number in the VOLUME parameter.
- Refer to Figure 24 for parameters that are mutually exclusive with DISP.

Examples of the DISP Parameter

```
//DD2       DD      DSNAME=FIX,UNIT=2400-1,VOLUME=SER=44889,
//                  DISP=(OLD,,DELETE)
```

Defines an existing data set and implies that the data set is to be kept if the step terminates normally. (For an existing data set, the system assumes it is to keep the data set if no disposition is specified.) The statement requests that the system delete the data set if the step abnormally terminates.

```
//STEP1     EXEC    PGM=FILL
//DD1       DD      DSNAME=SWITCH.LEVEL18.GROUP12,UNIT=2314,
//                  VOLUME=SER=LOCAT3,SPACE=(TRK,(80,15)),DISP=(,PASS)
//STEP2     EXEC    PGM=CHAR
//DD2       DD      DSNAME=XTRA,DISP=OLD
//DD3       DD      DSNAME=*.STEP1.DD1,DISP=(OLD,PASS,DELETE)
//STEP3     EXEC    PGM=TERM
//DD4       DD      DSNAME=*.STEP2.DD3,DISP=(OLD,CATLG,DELETE)
```

The DD statement named DD1 in STEP1 defines a new data set and requests that the data set be passed. If STEP1 abnormally terminates, the data set is deleted since it is a new data set and a conditional disposition was not specified. The DD statement named DD3 in STEP2 receives the passed data set and requests that the data set be passed. If STEP2 abnormally terminates, the data set is deleted because of the conditional disposition of DELETE. The DD statement named DD4 in STEP3 receives the passed data set and requests that the data set be cataloged at the end of the step. If STEP3 abnormally terminates, the data set is deleted because of the conditional disposition of DELETE.

The DLM Parameter—*keyword, optional*

The DLM parameter allows you to use a delimiter other than /* to terminate data defined in the input stream. By assigning a different delimiter in the DLM parameter, you can include a standard delimiter (/*) as data in the input stream.

```
DLM=delimiter
```

delimiter
specifies two characters that indicates the end of a group of data in the input stream.

Default: */**

Note: If there is a /*EOF JES2 statement in the jobstream, it acts as a delimiter regardless of the DLM value.

Rules for Coding

- For JES2, if one character is specified, the default is used. If more than 2 characters are specified, the job is flushed. For JES3, if an incorrect number of characters is coded, the job is flushed.
- The delimiter can be any two characters.
- If the delimiter value specified on DLM=delimiter contains any special characters, enclose the value in apostrophes. If you include an ampersand or an apostrophe in the delimiter, you must code each ampersand or apostrophe as two consecutive ampersands or apostrophes. Any character that is neither alphameric nor national and does not appear in the special character table must be treated as if it were a special character; that is, it must be enclosed in apostrophes. Violating the special character coding rules produces unpredictable results.
- The DLM parameter has meaning only on statements defining data in the input stream (DD * and DD DATA statements). If DLM is specified on any other statement, a mutually exclusive error message is issued.
- If you do code the DLM parameter on a DD DATA statement, the characters you assign as delimiters override any delimiter implied by the DD DATA statement. You must terminate the data with the characters you assigned in the DLM parameter.
- If the system encounters an error on the DD statement before the DLM parameter, it does not recognize the value assigned as a delimiter. When the card reader is emptied, the input reader device also causes the system to end an input data set.
- JES2 statements are not recognized if they are in an input stream between the DLM parameter and the nonstandard delimiter.

Example of the DLM Parameter

```
//DD1       DD      *,DLM=AA
            .
            .
            .
            data
            .
            .
AA
```

The DLM parameter assigns the characters AA as the valid delimiter for the data defined in the input stream by 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.

The DSID Parameter—*keyword, optional*

The DSID parameter specifies the data set identifier of an input or output data set on diskette for the 3540 diskette reader or writer utilities. Output data sets to be written to a 3540 diskette must be assigned to an output class that is processed by the diskette writer (an external writer). For the diskette writer to receive data sets, reserved classes for diskette output must be defined. To write data sets on a diskette, the operator must start the diskette writer to a 3540 device.

For more information about associated data sets, refer to the section, "Associated Data Sets (3540 Diskette)" in this book and to **OS/VS2 IBM 3540 Programmer's Reference.**


```
DSID=( id,[V] )
```

id
specifies the data set identifier. The identifier can be 1 - 8 characters in length. The characters must be alphameric, national, minus (hyphen), or left bracket (12-0 punch). The first character must be alphabetic or national.

V
specifies that the data set label must have been previously verified on a 3741 data entry terminal. (SYSIN only)

Rules for Coding

- Parentheses are not needed if only the id is coded.
- DSID on the DD * or DD DATA statement is ignored except when the JCL is processed by a diskette reader.
- Along with DSID, you can specify volume serial and logical record length information on the DD * and DD DATA statements.
- Refer to Figure 24 for parameters that are mutually exclusive with DSID. DSID can be specified with the DD *, DD DATA, and DD SYSOUT parameters; otherwise, it is ignored.

Example of the DSID Parameter

```
//JOB1      JOB       ,,MSGLEVEL=(1,1)
//STEP      EXEC      PGM=AION
//SYSIN     DD        *,DSID=(ABLE,V),VOL=SER=123456,
//                    DCB=LRECL=80
//SYSPRINT  DD        SYSOUT=E,DCB=LRECL=128,DSID=BAKER
```

The input is found on diskette 123456 in data set ABLE and must have been verified. The output will be created on diskette in data set BAKER.

The DSNAME Parameter—*keyword, optional*

The DSNAME parameter specifies the name of a data set. For new data sets, the name specified is assigned to the data set; for existing data sets, the system uses the name to locate the data set on the volume.

For further information on indexed sequential data sets and generation data groups, see, "Special Data Sets.". Partitioned data sets are described in **OS/VS2 MVS Data Management Services Guide.**

```
{DSNAME} = {dsname                                  }
{DSN   }   {dsname(member name)                     }
           {dsname(generation number)               }
           {dsname(area name)                       }
           {&&dsname                                }
           {&&dsname(member name)                   }
           {&&dsname(area name)                     }
           {*.ddname                                }
           {*.stepname.ddname                       }
           {*.stepname.procstepname.ddname          }
```

dsname
specifies a data set name.

dsname(member name)
specifies a nontemporary partitioned data set name and the name of a member within that data set.

dsname(generation number)
specifies the name of a generation data group (GDG) and the generation number (zero or a signed integer) of a generation data set within the GDG.

dsname(area name)
specifies the name of a nontemporary indexed sequential data set and an area of that data set (INDEX, PRIME, or OVFLOW).

&&dsname
specifies the name of a temporary data set.

&&dsname(member name)
specifies the name of a temporary partitioned data set and a member within that data set.

&&dsname(area name)
specifies the name of a temporary indexed sequential data set and an area of that data set (INDEX, PRIME, or OVFLOW).

***.ddname**
specifies that the data set name is to be copied from an earlier DD statement in the job step.

***.stepname.ddname**
specifies that the data set name is to be copied from an earlier DD statement, ddname, which appears in an earlier step, stepname, in the same job.

***.stepname.procstepname.ddname**
specifies that the data set name is to be copied from an earlier DD statement in a cataloged procedure. Stepname is the name of the job step that calls the procedure, procstepname is the name of the procedure step that includes the named DD statement, and ddname is the name of the DD statement that contains the data set name.

General Rules for Coding

- The DSNAME parameter can be abbreviated DSN.
- If the data set name begins with a blank character, the system assigns the data set a temporary data set name. Blank characters at the end of a data set name are ignored.
- Refer to Figure 24 for parameters that are mutually exclusive with DSNAME.

Special Characters

- If a data set name includes special characters as part of the name (the characters do not have special significance to the system), you must enclose the name in apostrophes (5-8 punch). If one of the special characters is an apostrophe, identify it by coding two consecutive apostrophes, for example, DSNAME='DAYS''END'.
- If the only special character is a period or a hyphen, you need not enclose the data set name in apostrophes.
- The following special characters have significance to the system and must not be enclosed in apostrophes: ampersands coded to identify temporary data sets; parentheses enclosing the member name of a partitioned data set, the area name of an indexed sequential data set, or the generation number of a generation data set; and the asterisk, used in the backward reference.
- The data set name should not consist of the 44 special characters (X'04') created by the 12-4-9 multi-punch (or any operation that converts the value of characters to X'04').

Nontemporary Data Sets

You can assign a nontemporary data set either an **unqualified** or **qualified** name. An unqualified name consists of 1 to 8 characters. The first character must be an alphabetic or national (@,#,$) character; the remaining characters can be any alphameric or national characters, a hyphen, or a plus zero (12-0) punch. A qualified name consists of multiple names joined by periods. The rules for coding each name within a qualified name are the same as for coding an unqualified name. A qualified data set name can include as many as 44 characters, including periods, unless the data set is a generation data set. Qualified names of generation data groups cannot exceed 35 characters, including periods.

If a qualified data set name is for an output tape data set and is more than 17 characters long, only the rightmost 17 characters are written to the tape header label. For additional information, refer to **OS/VS Tape Labels**.

Temporary Data Sets

- You need not code the DSNAME parameter when defining a data set that is created and deleted within a job (a temporary data set). The system will generate a name for the data set.
- If you do code the DSNAME parameter, the data set name consists of 1 to 8 characters and is preceded by two ampersands (& &). The first character following the ampersands must be an alphabetic or national (@,#,$) character; the remaining characters can be any alphameric or national characters, a hyphen, or a plus-zero (12-0) punch. The system generates a qualified name for the temporary data set that begins with SYS and includes the jobname, the temporary name assigned in the DSNAME parameter, and other identifying characters.
- The operating system generates a qualified name for the temporary data set. If several jobs enter the interpreter at the same time and contain DD statements with the same temporary data set name or with no data set name, the qualified name generated by the system will not be unique.
- A single ampersand preceding a data set name in a cataloged or in-stream procedure normally 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 within the procedure, the name is treated as a temporary data set name.

Special Data Sets

- Coding DSNAME=NULLFILE has the same effect as coding the DUMMY parameter. (The DUMMY parameter and DSNAME=NULLFILE are described in the chapter "Special Data Sets".)
- An **indexed sequential data set** can be either temporary or nontemporary. If you use only one DD statement to define an indexed sequential data set, omit the area name or code PRIME for the area name; for example, DSNAME=dsname or DSNAME=dsname(PRIME). To retrieve an indexed sequential data set, code only the data set name and omit the area name.
- If you assign a qualified name to a **generation data group**, the qualified name cannot exceed 35 characters, including periods. To retrieve all generations of a generation data group, omit the relative generation number in the DSNAME parameter.

Examples of the DSNAME Parameter

```
//DD1           DD      DSNAME=ALPHA,DISP=( ,KEEP),
//                      UNIT=2400,VOLUME=SER=389984
```

Defines a new data set whose name is ALPHA. Later job steps or jobs may retrieve this data set by supplying the data set name in the DSNAME parameter, unit information in the UNIT parameter, and volume information in the VOLUME parameter.

```
//DD2           DD      DSNAME=PDS(PROG12),DISP=(OLD,KEEP),UNIT=2314,
//                      VOLUME=SER=882234
```

Retrieves member PROG12 from the partitioned data set named PDS.

```
//DD3           DD      DSNAME=&&WORK,UNIT=2400
```

Defines a temporary data set. Since the data set is to be deleted at the end of the job step, the DSNAME parameter can be omitted. However, it may be included to facilitate a later reference to a passed data set; for example, DSNAME=&&WORK,DISP=OLD, in which case you must add DISP=(,PASS) to DD3.

```
//STEP1     EXEC    PGM=CREATE
//DD4       DD      DSNAME=&&ISDATA(PRIME),DISP=( ,PASS),UNIT=(2314,2),
//                  SPACE=(CYL,(10,,2),,CONTIG),VOLUME=SER=33489,
//                  DCB=DSORG=IS
//STEP2     EXEC    PGM=OPER
//DD5       DD      DSNAME=*.STEP1.DD4,DISP=(OLD,DELETE)
```

The DD statement named DD4 in STEP1 defines a temporary indexed sequential data set whose name is ISDATA. This DD statement is used to define all of the areas of an indexed sequential data set. The DD statement named DD5 in STEP2 retrieves the data set by referring to the earlier DD statement that defines the data set. Since the temporary data set will be passed when it is defined in STEP1, STEP2 can retrieve the data set.

The DUMMY Parameter—*positional, optional*

The DUMMY parameter specifies 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.

For further information on the DUMMY parameter, see "Defining a Dummy Data Set."

```
//ddname        DD       DUMMY
```


Rules for Coding

- 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. The DDNAME and DUMMY parameters are mutually exclusive.
- Code the DCB parameter if you would code it for normal I/O operations or if the DCB information is not supplied in the DCB macro by the executing program.
- Code AMP=AMORG if you are using VSAM and specify DUMMY for a data set.
- If you used the DUMMY parameter to test a program, when you want input or output operations performed on the data set, replace the DD statement that contains the DUMMY parameter with a DD statement that contains all of the parameters required to define this data set.
- When you nullify a procedure DD statement that contains the DUMMY parameter by coding the DSNAME parameter on the overriding DD statement, be sure that the data set name is not NULLFILE. Assigning the name NULLFILE in the DSNAME parameter has the same effect as coding DUMMY.
- If you code the DUMMY parameter and also request an access method other than the basic sequential access method (BSAM), virtual storage access method (VSAM) or queued sequential access method (QSAM) to read or write the data set, or if the DUMMY parameter is coded and the access method of BDAM load mode (BSAM with DCB MACRF=WL) is requested, a programming error will occur.
- Besides bypassing input or output operations on a data set, the DUMMY parameter causes the UNIT, SPACE, and DISP parameters, when coded on the DD DUMMY statement, to be ignored; however, these parameters are checked for syntax.
- If you code DUMMY on a DD statement, you do not nullify symbolic parameters that appear on the DD statement being overridden. You must assign values to, or nullify the symbolic parameters, on the DD statement being overridden, as described in the section "Assigning Values to and Nullifying Symbolic Parameters". Any parameters coded on a DUMMY statement must be syntactically correct.
- Backward references: If you code DUMMY on a DD statement and a later DD statement in the same job refers to this DD statement when requesting unit affinity (UNIT=AFF=ddname) or volume affinity (VOLUME=REF=*.stepname.ddname), the data set defined on the later DD statement will be assigned a dummy status.
- Data sets concatenated to a DUMMY data set will also be treated as a DUMMY data set by the system, in that I/O operations are bypassed. Disposition processing is performed, and devices and storage are allocated.
- If you use DD DUMMY and either VOL=REF=dsname or DCB=dsname, the referenced dsname must be cataloged or passed or the job fails.

Examples of the DUMMY Parameter

```
//OUTPUT        DD      DUMMY,DSNAME=X.X.Z,UNIT=2314,
//                      SPACE=(TRK,(10,2)),DISP=(,CATLG)
```

Defines a dummy data set. The parameters coded with the DUMMY parameter are not used.

```
//IN          DD      DUMMY,DCB=(BLKSIZE=800,LRECL=400,RECFM=FB)
```

Defines a dummy data set. The DCB parameter supplies information that was not supplied in the DCB macro instruction for the data control block. Otherwise, abnormal termination may occur.

If you are calling a cataloged procedure that contains the following DD statement in STEP4,

```
//IN          DD      DUMMY,DSNAME=ELLN,DISP=OLD,VOL=SER=11257,UNIT=2314
```

you can nullify the effects of the DUMMY parameter by coding:

```
//STEP4.IN        DD      DSNAME=ELLN
```

If you are calling a cataloged procedure that contains the following DD statement in STEP1,

```
//TAB         DD      DSNAME=APP.LEV12,DISP=OLD
```

you can make this DD statement define a dummy data set by coding:

```
//STEP1.TAB       DD      DUMMY
```

If you are calling a cataloged procedure that contains the following DD statement in a procedure step named LOCK,

```
//MSGS        DD      SYSOUT=A
```

you can make this DD statement define a dummy data set by coding:

```
//LOCK.MSGS       DD      DUMMY
```

The DYNAM Parameter—*positional, optional*

The DYNAM parameter specifies that a resource can be held in anticipation of reuse.

For further information, see "Dynamically Allocating and Deallocating Data Sets."

```
//ddname        DD      DYNAM
```


Rules for Coding

- Do not code any other parameters with the DYNAM parameter.
- Do not use the DDNAME parameter to refer to a DD DYNAM statement.
- To nullify the DYNAM parameter in a cataloged procedure, code the SYSOUT or DSNAME parameter in the overriding DD statement, but do not use the DSNAME=NULLFILE.
- Coding DYNAM on DD statements that will require dynamic allocation no longer establishes this DD statement as a DUMMY request. Rather, the number of DYNAM requests are added to the DYNAMNBR value only to acquire a control value necessary to track the resources to be held for reuse.
- Do not use any type of DD parameter referback to a DD DYNAM statement.
- Do not code the DYNAM parameter on the first DD statement of a group of DD statements defining a data set concatenation.
- Do not code the DYNAM parameter on a DD statement having a ddname that is meaningful to the system; for example, JOBLIB, SYSCHK, etc.

Example of the DYNAM Parameter

```
//INPUT     DD      DYNAM
```

Specifies that the control value for dynamically allocated resources held for reuse is incremented by one for dynamic allocation.

The FCB Parameter—*keyword, optional*

The FCB parameter specifies the forms control image to be used to print an output data set on a 3800 or 3211 printer, or the data protection image to be used for the 3525 card punch or for SYSOUT.

For further information on the forms control buffer, see **OS/VS2 System Programming Library: Data Management, OS and OS/VS Programming Support for the IBM 3505 Card Reader and IBM 3525 Card Punch,** or **IBM 3800 Printing Subsystem Programmer's Guide.**

```
FCB=(image-id  [,ALIGN ]  )
               [,VERIFY]
```

image-id
specifies 1-4 alphameric or national characters that identify the image or module to be loaded into the forms control buffer.

ALIGN
requests that the operator check the alignment of the printer forms before the data set is printed.

VERIFY
requests that the operator verify that the image displayed on the printer is the desired one. The operator is also given an opportunity to align the printer forms.

Default: For the 3211, the image currently in the buffer. If one is not there, the operator is requested to specify an image. For the 3800, the machine default is 6 lines per inch for the form size that is on the printer. For JES2, the buffer value must have a default flag. For JES3, the FCB parameter defaults to a system standard or job class installation-defined default.

General Rules for Coding

- The ALIGN and VERIFY subparameters are ignored for SYSOUT data sets.
- The ALIGN subparameter is not used by the 3800.
- Parentheses are not needed when only the image-id is specified.
- The FCB parameter is ignored for SYSOUT data sets on the 3525. JES2 and JES3 use it to request a carriage tape for a non-FCB printer or to load the FCB on a printer having the FCB feature.
- For a work station that does not use a peripheral data set information record (PDIR), add an FCB image to SYS1.IMAGELIB. At setup time, JES3 translates the FCB into a set vertical format (SVF).
- For a work station that uses a PDIR, the FCB image is defined in the work station. The image-id specified as an FCB parameter on the DD statement must be the name of the FCB defined in the PDIR work station.
- STD1 and STD2 should not be used as image-ids for SYSOUT unless specified by your installation.
- Refer to Figure 24 for parameters that are mutually exclusive with FCB. FCB is also mutually exclusive with the DCB subparameters RKP, CYLOFL, INTVL, and FRID.

Using the 3800 Printing Subsystem to Print Dumps with More Data per Page

In order to request dump output at 8 lines per inch, specify FCB=STD3 on the dump-related DD statement. You can also request a dump of 204 characters per line by coding DUMP on the CHARS parameter of the dump-related DD statement.

Examples of the FCB Parameter

```
//DD1           DD      UNIT=3211,FCB=(IMG1,VERIFY)
```

Defines the output data set that is to be written to a 3211 printer. The FCB parameter requests that the data set be written using the control information corresponding to the forms control image with the code IMG1. Since VERIFY is coded, the forms control image is displayed on the printer before the data set is printed.

```
//DD2           DD      SYSOUT=A,FCB=IMG2
```

If output class A routes output to a printer having the forms control buffer feature, JES2 loads the image identified by 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 specified carriage tape (in this case, IMG2) on the printer.

```
//OUTPUT        DD      UNIT=3211,FCB=(6,ALIGN)
```

Requests that the operator check the alignment of the printer forms before the data set is printed.

```
//PUNCH         DD      UNIT=3525,FCB=DP2
```

The unit specification is for the 3525 card reader. Therefore, the FCB parameter is defining the data protection image to be used for the 3525.

```
//SYSUDUMP      DD      SYSOUT=A,FCB=STD3
```

Requests that the 3800 print a dump at 8 lines per inch.

The FLASH Parameter — *keyword, optional*

The FLASH parameter identifies the forms overlay to be used on the 3800 Printing Subsystem and, optionally, specifies the number of copies on which the forms overlay is to be printed.

For information on designing and making or obtaining forms overlays, see the **Forms Design Reference Guide for the IBM 3800 Printing Subsystem.**

```
FLASH=(overlay name[,count])
```

overlay name
identifies the forms overlay frame that the operator is to insert into the printer before printing begins (1-4 alphameric or national characters). There is no system verification that the operator inserted the correct frame.

count
indicates the number of copies (from 1 to 255) to be flashed with the overlay, beginning with the first copy printed.

Default for count: 255

If 0 is specified for count, the default is assumed.

General Rules for Coding

- The FLASH parameter can be specified on a SYSOUT DD statement when the 3800 is allocated as a system output device or on an output DD statement when the 3800 is allocated as a direct output device.
- The overlay name cannot be omitted. The count subparameter is optional, but a null position is not allowed. (For example, FLASH=(ABCD,) is invalid.)
- Parentheses are not needed if count is omitted.
- FLASH can also be coded on the JES3 FORMAT PR statement or the JES2 OUTPUT statement.
- Refer to Figure 24 for parameters that are mutually exclusive with FLASH.

Rules for Coding Count

- The maximum number of copies that can be flashed with the forms overlay is the value of nnn or the sum total of group values on the COPIES parameter. If count is greater than this value, the difference is ignored.
- If count is not specified or coded as 0, all copies printed are flashed with the specified overlay.

Example of the FLASH Parameter

```
//DD1    DD       SYSOUT=A,COPIES=10,FLASH=(ABCD,5)
```

The operator receives a message to insert the forms overlay frame named ABCD into the printer. The first 5 copies of the data set printed are flashed with the forms overlay.

The FREE Parameter—*keyword, optional*

The FREE parameter specifies when the data set defined by the DD statement is to be deallocated; it may be deallocated either at the end of the step or when the data set is closed.

Code CLOSE whenever you don't want to monopolize resources — for example, devices, volumes, exclusive access rights to a data set—any longer than necessary.

```
FREE=  {END  }
       {CLOSE}
```


END
specifies that the data set is to be deallocated at the end of the step.

CLOSE
specifies that the data set is to deallocated at the time it is closed.

Default: END

If the value is incorrectly coded, the default value is substituted and a warning message is issued.

Rules for Coding

- Code the FREE=CLOSE parameter on a SYSOUT DD statement to cause JES2 and JES3 to spin off the data set.
- FREE=CLOSE should not be specified for a data set that is opened and closed more than once during a job step. If the data set is reopened, the job step will abnormally terminate unless there is an intervening dynamic allocation.
- FREE=CLOSE is ignored for a DD statement that has a ddname of JOBCAT, JOBLIB, STEPCAT, or STEPLIB; for a data set that is a member of a concatenated group; or for a task that abnormally terminates.
- When FREE=CLOSE is coded and the data set is closed with the LEAVE option specified on the CLOSE macro instruction, the data set is not deallocated until the end of the job step.
- When coding FREE=END, deallocation can occur at the end of the job or at the end of the job step. See the following examples:

Example 1.

```
//STEP1         EXEC    PGM=ABLE1
//DD1           DD      DSN=A,DISP=( ,PASS ),FREE=END
//STEP2         EXEC    PGM=ABLE2
//DD2           DD      DSN=A,DISP=( OLD,CATLG ),FREE=END
```

In this example, data set A is received in STEP2 and, at the termination of STEP2, it will be cataloged and deallocated. FREE=END does not cause data set A to be deallocated at the end of step 1 because a disposition of pass has been specified.

Example 2.

```
//STEP1         EXEC    PGM=BAKER1
//DD            DD      DSN=A,DISP=( NEW,PASS ),FREE=END
//STEP2         EXEC    PGM=BAKER2
/*
```

In this example, data set A is a new, passed, unreceived data set and will not be deallocated until job termination.

- Refer to Figure 24 for parameters that are mutually exclusive with FREE.

Example of the FREE Parameter

```
//EA33      DD      SYSOUT=D,FREE=CLOSE
```

The data set allocated to class D will be deallocated and spun off (available for printing) when the data set is closed rather than at the end of the job.

```
//EA33      DD      DSN=SYBIL,DISP=OLD,FREE=CLOSE
```

The data set is dequeued when deallocated and available for someone else to use.

The HOLD Parameter—*keyword, optional*

The HOLD parameter specifies that an output data set is to be held on a queue until released by a central or remote operator at the target destination, or by the time-sharing user who is eligible to free the data set. If you are receiving the output at the destination (work station) you should inform either the central operator or the work station to which the output will be sent to release the data set for processing.

For further information on the HOLD parameter, see "Obtaining Output" for either JES2 or JES3.


```
HOLD= {YES}
      {NO }
```

YES
specifies that processing of the output data set is to be deferred until the data set is released.

NO
specifies that processing of the output is to be determined by the installation-defined specification associated with the output class. This specification can be overridden by using the unallocation verb of dynamic allocation or the TSO FREE command.

Default: NO

If an incorrect value is coded, the default is assumed and a warning message is issued. The job continues.

Rule for Coding

The HOLD parameter can be coded only on a DD statement that includes the SYSOUT parameter. Otherwise, it is ignored.

Example of the HOLD Parameter

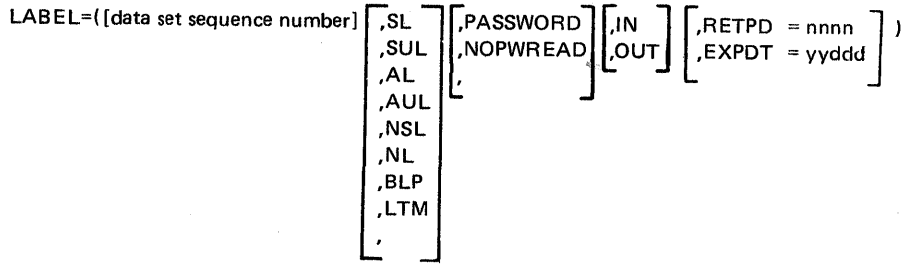
```
//JOB01     JOB     ,'HAROLD DUQUETTE',MSGLEVEL=1
//STEP1     EXEC    PGM=MJCOSCO
//DD1       DD      SYSOUT=B,DEST=RMT6,HOLD=YES
```

The output from JOB01 is held on a queue until the user identified by RMT6 or the central or remote operator requests that the data set be released.

The LABEL Parameter—*keyword, optional*

The LABEL parameter indicates the type and contents of the label or labels associated with a data set.

For detailed information on tape label definitions and processing, see **OS/VS Tape Labels**; labels on direct access devices are described in **OS/VS2 MVS Data Management Services Guide.** A detailed description of protecting a data set by assigning a password is included in **OS/VS2 System Programming Library: Data Management.**



data set sequence number
specifies the relative position of a data set on a tape volume where processing is to begin.

SL
specifies that a data set has IBM standard labels.

SUL
specifies that a data set has both IBM standard and user labels.

AL
specifies that a tape data set has American National standard labels.

AUL
specifies that a tape data set has American National standard and user labels.

NSL
specifies that a tape data set has nonstandard labels.

NL
specifies that a tape data set has no labels.

BLP
specifies that the system is to bypass label processing for a tape data set.

LTM
specifies that the data set can have a leading tapemark.

PASSWORD
specifies that a data set cannot be read, changed, deleted, or written to unless the operator or time-sharing user supplies the correct password.

NOPWREAD
specifies that a data set cannot be changed, deleted, or written to unless the operator or time-sharing user supplies the correct password. No password is necessary for reading the data set.

IN
specifies that a BSAM or BDAM data set is to be processed for input only. This subparameter overrides the INOUT (BSAM) option or UPDAT (BDAM) option in the OPEN macro instruction.

OUT
specifies that a BSAM data set is to be processed for output only. This subparameter overrides the OUTIN option in the OPEN macro instruction.

RETPD=nnnn
specifies the number of days that the data set must be kept before it can be deleted or written over by another data set. nnnn is a 1 to 4-digit number.
The RETPD value is converted to an expiration date (EXPDT) for the data set. The highest converted date possible is 99365. If you code an nnnn value, that when added to the current date, will exceed 99365, the data set will not be retained for the entire period specified. Therefore, any RETPD specified that extends to or beyond January 1,2000 is assigned the EXPDT default value of 99365. The EXPDT default value is calculated on a 365-day year, ignoring leap years. (Note that when coding EXPDT you may code 366.)

EXPDT=yyddd
Assign a 2-digit year number and a 3-digit day number. The three-digit day number can not be less than 001 and can not exceed 366. For example, February 2, 1973 would be specified as 73033.

General Rules for Coding

- All the subparameters except the last subparameter in the LABEL parameter are positional. Therefore, if you code one subparameter and omit a previous subparameter, indicate its absence with a comma.
- If you only want to specify the data set sequence number, RETPD, or EXPDT, you can omit the parentheses and code LABEL=data set sequence number, LABEL=RETPD=nnnn, or LABEL=EXPDT=yyddd.
- Refer to Figure 24 for parameters that are mutually exclusive with LABEL.
- Do not specify both the DCB FUNC subparameter and the LABEL parameter; unpredictable results will occur.

Rules for Coding Data Set Sequence Number

- Default: If you omit this subparameter or code 0, the system assumes that this is the first data set on the tape volume, unless the data set is passed or cataloged. If a data set is cataloged, the system obtains the data set sequence number from the catalog. The data set sequence number for a passed data set is obtained from the passing step.
- The system ignores a data set sequence number coded for a passed data set and uses the data set sequence number obtained from the passing step.
- For a multivolume data set the data set sequence number is not used on the second and subsequent volumes that are processed.
- The data set sequence number is a 1- to 4-digit number.
- The data set sequence number is always retrieved from the catalog for GDG ALL requests.

Rules for Coding Label Types

- Default: If you omit this subparameter, the system assumes that the data set has only IBM standard labels (SL).
- Code a comma when the label type subparameter is omitted and another positional subparameter follows.
- Data sets on direct access devices always have standard labels; they can optionally have user labels also. Therefore, only SL or SUL can be coded for data sets on direct access devices. SUL cannot be coded for partitioned or indexed sequential data sets.
- Label type information is not retained for cataloged data sets. You must code the LABEL parameter and specify label type if you refer to a cataloged data set that does not have IBM standard labels only.
- If the system does not have the bypass-label-processing (BLP) feature, specifying BLP has the same effect as specifying NL.
- If you specify BLP and the tape volume has labels, a tapemark delimits the data set. For a tape with labels to be positioned properly, the data set sequence number subparameter must

be coded and must reflect all labels and data sets that precede the desired data set. Do not, however, specify BLP for GDG All data sets. Since the system obtains the data set sequence

number from the catalog for GDG ALL data sets, specifying BLP might result in incorrect tape positioning.

- If you are processing ASCII data on unlabeled tapes (NL), you must code OPTCD=Q in the DCB macro instruction or in the DCB parameter on the DD statement.
- Direct access devices used when referring the system to an earlier volume request, obtain label type information from the LABEL parameter specified in the DD statement and not from the source you refer it to.

Rules for Coding PASSWORD and NOPWREAD (no-password-read)

- Only data sets with IBM or American National standard labels can be password-protected.
- When specifying password protection for an existing data set by coding PASSWORD or NOPWREAD, you must open the data set for output processing the first time it is used during the job step.
- When specifying PASSWORD or NOPWREAD for a data set, a password must be assigned to that data set in the PASSWORD data set. If a password is not assigned, attempts to open that data set for output (if NOPWREAD is coded) or for input or output (if PASSWORD is coded) result in abnormal termination.
- Code a comma when this subparameter is omitted and ,IN or ,OUT follows. For a new data set, the data set is not password-protected.
- You might not be able to password-protect data sets with data set names that are longer than 17 characters. MVS retains, in the tape label, only the rightmost 17 characters of the data set name. As a result, two or more names could look identical for purposes of password checking. For example, the following names would appear as identical:

```
PDQ.USERDATA.EXAMPLE1
XYZ.USERDATA.EXAMPLE1
```

Rule for Coding IN and OUT

When the IN subparameter is coded, any attempt by the processing program to process the data set for output results in abnormal termination. If OUT is coded and the processing program attempts to process the data set for input, the step is abnormally terminated.

Rules for Coding RETPD and EXPDT

- A data set for which RETPD or EXPDT is specified is deleted by the creating step when a disposition of DELETE is specified.
  - Specifying a normal disposition of DELETE causes the dat set to be deleted when the creating step terminates normally.
  - Specifying a conditional disposition of DELETE causes the data set to be deleted if the creating step abnormally terminates.
- To delete a data set before the expiration date or retention period has passed, use one of the following:
  - The DELETE command, as described in **OS/VS Access Method Services**, to delete data sets cataloged in a VSAM catalog
  - The IEHPROGM utility, as described in **OS/VS2 MVS Utilities**, to delete data sets that are not cataloged in a VSAM catalog
  - The SCRATCH macro with the OVRD parameter, as described in **OS/VS2 System Programming Library: Data Management**, to delete the data set control block, which makes the space occupied by that data set available for reallocation.
- Do not specify or imply RETPD or EXPDT for a temporary data set.

Note: The RETPD and EXPDT parameters should not be used to protect valuable data. Instead use password protection or the Resource Access Control Facility (Program Product, Program No. 5740-XXH) for data protection.

Examples of the LABEL Parameter

```
//DD1       DD      DSNAME=HERBI,DISP=(NEW,KEEP),UNIT=TAPE,
//                  VOLUME=SER=T2,LABEL=(3,NSL,RETPD=188)
```

Defines a new data set. The LABEL parameter tells the system: (1) this data set is to be the third data set on the tape volume; (2) this tape volume has nonstandard labels; (3) this data set is to be kept for 188 days.

```
//DD2       DD      DSNAME=A.B.C,DISP=(,CATLG,DELETE),UNIT=2400-2,
//                  LABEL=(,NL)
```

Defines a new data set and requests that the system catalog it. The catalog entry for this data set will not indicate that the data set has no labels. Therefore, each time this data set is referred to by a DD statement, the statement must include LABEL=(,NL).

```
//DD3       DD      DSNAME=SPECS,UNIT=2400,VOLUME=SER=10222,
//                  DISP=OLD,LABEL=4
```

Defines an existing data set. The LABEL parameter indicates that the data set is the fourth data set on the tape volume.

```
//STEP1     EXEC    PGM=FIV
//DDX       DD      DSNAME=CLEAR,DISP=(OLD,PASS),UNIT=2400-4,
//                  VOLUME=SER=1257,LABEL=(,NSL)
//STEP2     EXEC    PGM=BOS
//DDY       DD      DSNAME=*.STEP1.DDX,DISP=OLD,LABEL=(,NSL)
```

The DD statement named DDX in STEP1 defines an existing data set that has nonstandard labels and requests that the system pass the data set. The DD statement named DDY in STEP2 receives the passed data set. Unit and volume information is not specified since this information is available to the system; the label type is not available to the system and must be coded.

The MODIFY Parameter — *keyword, optional*

The MODIFY parameter specifies the name of a copy modification module to be loaded into the 3800 Printing Subsystem. This module contains predefined data such as legends, column headings, or blanks, and specifies where and on which copies the data is to be printed. The module is defined and stored on SYS1.IMAGELIB using the IEBIMAGE utility program.

For further information on the copy modification module and the IEBIMAGE utility program, see the **IBM 3800 Printing Subsystem Programmer's Guide.**

```
MODIFY=(module name[,trc])
```

module name
identifies a copy modification module previously stored in SYS1.IMAGELIB to be used to replace blanks or data in the printed data set. module name is 1-4 alphameric or national characters.

trc
indicates the table reference character (0-3) that corresponds to a character arrangement table specified with the CHARS parameter. This table is used for printing of the copy modification data.

Default for trc: 0

Rules for Coding

- The MODIFY parameter can be specified on a SYSOUT DD statement when the 3800 is allocated as a system output device or on an output DD statement when the 3800 is allocated as a direct output device.
- The trc subparameter can be coded as 0, 1, 2, or 3. The value corresponds to the sequence in which the character arrangement tables are specified with the CHARS parameter, (for example, 1 refers to the second table name coded with the CHARS parameter).
- The module name cannot be omitted. The trc subparameter is optional, but a null position is not allowed. (For example, MODIFY=(A,) is invalid.)
- Parentheses are not needed if trc is omitted.
- MODIFY can also be coded on the JES3 FORMAT PR statement or the JES2 OUTPUT statement.
- Refer to Figure 24 for parameters that are mutually exclusive with MODIFY.

Example of the MODIFY Parameter

```
//DD1   DD      UNIT=3800,MODIFY=(A,0),CHARS=(GS15,GS10)
```

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 the positions in the data set to be replaced, and the copies that are to be modified. The first character arrangement table coded with the CHARS parameter, GS15, is used to determine the character arrangements.

The MSVGP Parameter—*keyword, optional*

The MSVGP parameter specifies the identification of a group of mass storage volumes that reside on a mass storage system (MSS) device.

```
MSVGP=( id[,ddname] )
```

id
indicates a 1-8 alphameric or national character identifier (in any order) that defines the mass storage volume group.

ddname
identifies the name of a DD statement that appears earlier in the job step. Code this parameter when you want the data set described by the DD statement named 'ddname', and the data set described by the DD statement on which this parameter is coded, to be allocated to different volumes. The ddname parameter can only be used if MSS Enhancements (SU24) has been installed.

Rules for Coding

- MSVGP is ignored for specific volume requests. The following rules apply only to nonspecific volume requests.
- MSVGP is mutually exclusive with ABSTR, MXIG, and ALX subparameters of the SPACE parameter and the SER subparameter of the VOLUME parameter.
- Refer to Figure 24 for parameters that are mutually exclusive with MSVGP.
- Positional parameters related to the VOLUME parameter can be specified with MSVGP.
- The SPACE parameter is not required when MSVGP is used on nonspecific requests except for BPAM and ISAM data organization.
- The unit count in the UNIT parameter must be less than the volume count in the VOLUME parameter to guarantee allocation of a non-sharable unit.
- MSVGP results in a private volume for nonspecific volumes (even if a non-MSS volume is allocated to the request.) Therefore, coding VOL=PRIVATE is redundant.
- For a new, nonspecific, permanent data set request where MSVGP is not specified, a mounted 3330V storage volume is used, if one exists. If one does not exist, a volume is selected from SYSGROUP. If a volume is selected from SYSGROUP, the SPACE parameter must be coded or the job is failed.
- For a new, nonspecific, temporary data set request where MSVGP is not specified, a mounted 3330V public or storage volume is used, if one exists. If one does not exist or there is not enough space, a volume is selected from SYSGROUP.
- To guarantee allocation to SYSGROUP for nonspecific requests, specify MSVGP=SYSGROUP.
- The ddname parameter must refer to a DD statement that appears earlier in the job step.
- If you code the ddname parameter, you must also code the id parameter.
- If volume separation within the group is not possible, the job is failed.
- Only one ddname can be coded.
- For concatenated DD's (including GDG ALL's), ddname applies to only the first DD of the concatenation.

Examples of the MSVGP Parameter

```
//DD1    DD  DSN=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 has previously defined such a group using a mass storage system service and has assigned at least three volumes to this group.) Assume that this service also provides a space default of SPACE=(CYL,(200,100)). The system selects a volume from the group that contains at least 200 cylinders of space.

During step execution, if more than 200 cylinders are required, end of volume is entered. If 100 more cylinders are not available on the mounted volume, it is dismounted. The system again selects a volume from group AB$1234[that has at least 100 cylinders and causes this volume to be mounted. A volume count of three will allow the data set to extend over up to three volumes. If more are required, the step abnormally terminates.

```
//DD1    DD  DSN=MASTRIN,DISP=OLD
//DD2    DD  DSN=MASTROUT,UNIT=3330V,DISP=(,CATLG),
//           MSVGP=(AB$1234@,DD1)
```

DD1 specifies an existing cataloged data set. DD2 describes a new data set that will be allocated to a volume belonging to the mass storage volume group AB1234@.

Since DD1 is specified as the ddname parameter of the MSVGP keyword, that is coded on DD2, the system will ensure that the DD2 data set, MASTROUT, is not allocated to the same volume(s) that contain the DD1 data set, MASTRIN.

The OUTLIM Parameter—*keyword, optional*

The OUTLIM parameter specifies a limit for the number of logical records you want included in the output data set being routed through the SYSOUT data set. When the limit is reached, an exit may be taken to a user-supplied routine that determines whether to cancel the job or increase the limit. If the exit routine is not supplied, the job is terminated. For more information, see **OS/VS2 System Programming Library: System Management Facilities (SMF)**.

```
OUTLIM=number
```


number
specifies a number between 1 and 16777215, indicating the maximum number of logical records to be included in the output data set being routed through the output stream.

Default: For JES2, there is no default for output limiting. JES3 uses an installation-defined default; the default applies if you do not code OUTLIM or if you code OUTLIM=0.

Rules for Coding

- The OUTLIM parameter is ignored unless SYSOUT is coded in the operand field of the same DD statement.
- For SYSABEND and SYSUDUMP DD statements, JES3 ignores the OUTLIM parameter.
- If you code OUTLIM, do not code the DCB parameters CPRI or THRESH, or your OUTLIM value may be altered.
- Refer to Figure 24 for parameters that are mutually exclusive with OUTLIM.

Example of the OUTLIM Parameter

```
//OUTPUT    DD      SYSOUT=F,OUTLIM=1000
```

The limit for the number of logical records is 1000.

The PROTECT Parameter —*keyword, optional*

The PROTECT parameter requests RACF protection for either a direct access data set or a tape volume. IBM non-standard label tapes (NSL), IBM standard label tapes (SL, SUL), and ANSI label tapes (AL,AUL) may be protected when opened for output. Direct access data sets may be protected via the PROTECT parameter only when they are created.

The PROTECT parameter can be used only if System Security Support (SU32) has been installed and RACF (Program Product, Program No. 5740-XXH) is installed and active

```
PROTECT=YES
```

Yes
Requests RACF protection for either a direct access data set or for a tape volume.

Rules for Coding

- For Tape Volumes
  - The volume must have a use attribute of PRIVATE either explicitly or implicitly defined.
  - The label specification must be either SL, SUL, AL, AUL or NSL.
  - With the exception of NSL tapes, the data set named on the DD statement must be the first file on the tape.
  - With the exception of NSL tapes, multi-volume data sets must have a volume sequence number of one.
  - The data set must be opened for either OUTPUT or OUTIN.
  - The data set status must not be MOD treated as OLD.
  - In order to process NSL requests, the NSL installation exit routine must issue a RACFDEF macro instruction. See **OS/VS2 System Programming Library: Supervisor** for a description of RACDEF.
- For Direct Access Data Sets
  - The data set cannot have a temporary data set name.
  - The data set status must be NEW or MOD treated as NEW.
  - The data set must have a disposition of other than DELETE for either normal or abnormal termination.
- Refer to figure 24 for parameters that are mutually exclusive with the PROTECT parameter.

Examples of the PROTECT Parameter

```
//DASD      DD      UNIT=3330,VOL=SER=333000,DSN=USERID.MYDATA,
//                  DISP=(,CATLG),SPACE=(TRK,1),PROTECT=YES
```

Requests RACF protection for the newly created direct access data set USERID.MYDATA.

```
//TAPEDATA  DD      UNIT=2400,DISP=(OLD,UNCATLG),
//                  LABEL=(,SL),VOL=SER=T49850,DSN=TAPEDS,
//                  PROTECT=YES
```

Requests RACF protection for tape volume T49850. Because a specific tape volume is requested, it is automatically given the PRIVATE attribute. The volume has a standard label, and the data set TAPEDS must be the first file on the tape.

The QNAME Parameter—*keyword, optional*

The QNAME parameter allows you access to messages received through TCAM for processing by an application program.

```
QNAME=process-name[.tcamname]
```

process-name
specifies up to eight alphameric or national characters for the name of a TPROCESS macro instruction that defines a destination queue for messages that are to be processed by an application program. (The first character must be alphabetic or national).

tcamname
specifies the name of a TCAM job or started task (using the START command) that consists of from 1 to 8 alphanumeric characters. The first character must be alphabetic or national.

Rules for Coding

- The process-name must be identical to the symbolic name on the TPROCESS macro.
- The DCB parameter is the only parameter that can be coded on a DD statement with the QNAME parameter. BLKSIZE, BUFL, LRECL, OPTCD, and RECFM are the only operands that can be specified as subparameters.

Example of the QNAME Parameter

```
//DYD       DD       QNAME=FIRST,DCB=(RECFM=F,LRECL=80,BLKSIZE=320)
```

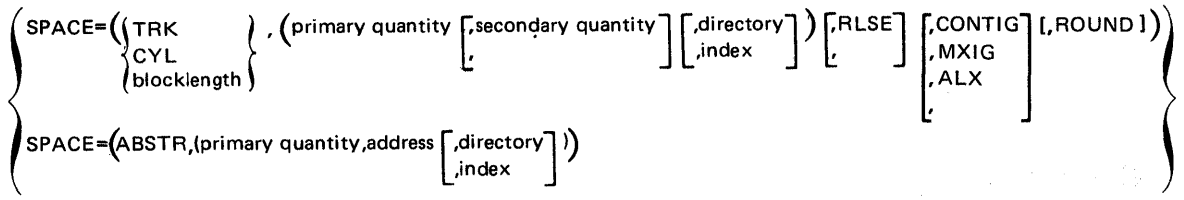
Used in an application program to define data that is used by TCAM. "FIRST" is the name of the TPROCESS macro that specifies the destination queue through which messages that must be processed by the application program is to be routed. The DCB parameter supplies information that was not supplied in the DCB macro instruction for the data control block.

```
//DXD       DD       QNAME=SECOND.TCAM01
```

This is an example of coding QNAME in an application program that defines data that TCAM01 uses. "SECOND" is the name of the TPROCESS macro that specifies a destination queue.

The SPACE Parameter—*keyword, optional*

The SPACE parameter indicates how much space should be allocated on a direct access volume for a new data set.



TRK
specifies that space is to be allocated by track.

CYL
specifies that space is to be allocated by cylinder.

block length
specifies the average block length of the data. The system computes how many tracks to allocate.

primary quantity
specifies how many tracks or cylinders are to be allocated, or how many blocks of data are to be contained in the data set.

secondary quantity
specifies how many more tracks or cylinders are to be allocated if additional space is required. This allocation can be done up to 16 times for each volume, less the number of extents for primary quantity and user-label space (BDAM data sets cannot be extended.)

directory
specifies the number of 256-byte records that are to be contained in the directory of a partitioned data set.

index
specifies how many tracks or cylinders are required for the index of an indexed sequential data set. The number of tracks must equal one or more cylinders.

RLSE
specifies that space allocated to an output data set that is not used when the data set is closed, is to be released.

CONTIG
specifies that space allocated to the data set must be contiguous. This subparameter applies only to the primary space allocation.

MXIG
specifies that the space allocated to the data set must be the largest area of contiguous space on the volume, and the space must be equal to or greater than the space requested. This subparameter applies only to the primary space allocation.

ALX
specifies that up to five different contiguous areas of space are to be allocated to the data set and each area must be equal to or greater than the space requested. This subparameter applies only to the primary space allocation.

ROUND
specifies that space is requested by specifying the average block length of the data and that the space allocated to the data set must be equal to an integral number of cylinders.

ABSTR
specifies that the data set is to be placed at a specific location on the volume.

primary quantity
specifies the number of tracks to be allocated to the data set.

address
specifies the track number of the first track to be allocated.

directory
specifies the number of 256-byte records in the directory of a partitioned data set.

index
specifies the number of tracks that are required for the index of an indexed sequential data set. The number of tracks must be equal to one or more cylinders.

General Rules for Coding

- 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.
- If you do not code secondary, directory, or index quantities, you need not enclose the primary quantity in parentheses.
- Code the second format of the SPACE parameter when you want a data set placed in a specific position on a direct access volume.
- Refer to Figure 24 for parameters that are mutually exclusive with SPACE.

Rules for coding when using mass storage volumes (new data sets only)

- The SPACE parameter must be coded when VOL=SER is coded. It is optional when MSVGP is coded. If you do not code VOL=SER or MSVGP, SPACE must be coded whether or not you code VOL=PRIVATE.
- Contiguous space is the MSVGP default. If you want non-contiguous primary space allocation, you must specify the SPACE parameter.

Rules for Coding Blocklength

- When you request space in units of blocks, the average blocklength cannot exceed 65,535.
- If the blocks have keys, code the DCB subparameter KEYLEN on the DD statement and specify the key length.

Rules for Coding Primary Quantity

- There must be enough available space on one volume to satisfy the primary quantity. If you request a particular volume and there is not enough space available on the volume to satisfy your request, the job step is terminated. You must consider track overflow when computing track requirements.
- When specifying tracks and cylinders, the primary quantity includes the number of tracks and cylinders assigned to the directory.

Rules for Coding Secondary Quantity

- Code a comma when this subparameter is omitted and the directory or index subparameter follows. The system does not allocate additional space if it is required.
- The system computes the number of tracks required for the secondary quantity based on what is specified in the DCB subparameter BLKSIZE, the DCB macro, or the SPACE parameter (average blocksize).
- If you do specify a secondary quantity and the data set requires additional space, the system allocates this space based on the quantity you specified. The system attempts to allocate the secondary quantity in contiguous tracks or cylinders. If contiguous space is not available, the

system attempts to allocate the secondary quantity in up to five noncontiguous blocks (extents) of space.

- Each time the data set requires more space, the system allocates the secondary quantity. This space is allocated on the same volume on which the primary quantity was allocated until: (1) there is not enough space available on the volume to allocate the secondary quantity, or (2) a total of 16 extents have been allocated to the data set. If either of these conditions is satisfied, the system must allocate the secondary quantity on another volume.
- If there is no more space available on those volumes that you requested, if at least one volume is demountable, the system will request that scratch (non-specific) volumes be mounted until the secondary allocation is complete. If there is no demountable volume, the job step will abnormally terminate.

Rule for Coding Directory

When creating a partitioned data set, you must request space for a directory.

Rules for Coding RLSE

- When coding RLSE for an existing data set, you must recode the unit of measurement and primary quantity exactly as they appeared in the original request.
  For example, if your original request was:

  ```
  SPACE=(TRK,(100,50))
  ```

  you can release unused tracks when you retrieve the data set by coding:

  ```
  SPACE=(TRK,(100),RLSE)
  ```

- Code a comma when RLSE is omitted and another subparameter follows.
- If you specify RLSE and an abnormal termination occurs, unused space is not released if the data set is open.
- The system ignores a request to release unused space when a data set is closed if:
  - Another job is allocated to the data set.
  - Another task in the same job is processing an OPEN, CLOSE, EOV, or FEOV request against the data set.
  - Another DCB is open against the data set.
- The RLSE subparameter is ignored when the TYPE=T option is coded in the CLOSE macro instruction.
- 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.

Rules for Coding MXIG, ALX, and CONTIG

- Code a comma when these subparameters are omitted and the ROUND subparameter follows.
- Do not code either the MXIG or ALX subparameters for an indexed sequential data set.
- If CONTIG is specified and contiguous space is not available, the job is terminated.
- If you code a secondary quantity and request contiguous space, the primary request will be satisfied with contiguous space, but the secondary quantity will not necessarily be contiguous.

Rule for Coding ROUND

- Do not code the ROUND subparameter for an indexed sequential data set.

Examples of the SPACE Parameter

```
//DD1       DD      DSNAME=&&TEMP,UNIT=MIXED,SPACE=(CYL,10)
```

Defines a temporary data set and requests that the system assign any available tape or direct access volume. (UNIT=MIXED specifies a user-assigned group name of units that consists of tape and direct access devices). If a tape volume is assigned, the SPACE parameter will be ignored; if a direct access volume is assigned, the SPACE parameter will be used to allocate space to the data set. The SPACE parameter includes only the required subparameters (that is, the type of units and a primary quantity), and requests that the system allocate 10 cylinders.

```
//DD2       DD      DSNAME=PDS12,DISP=(,KEEP),UNIT=2314,
//                  VOL=SER=25143,SPACE=(CYL,(10,,10),,CONTIG)
```

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.

```
//REQUEST   DD      DSNAME=PET,DISP=NEW,UNIT=3330,VOL=SER=606674,
//                  SPACE=(1024,(75)),DCB=KEYLEN=8
```

The average blocklength of the data is 1024 bytes and 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 what device is requested in the UNIT parameter.

The SUBSYS Parameter—*keyword, optional*

The SUBSYS parameter requests that the specified subsystem process the data set defined by the DD statement on which SUBSYS is coded. Optionally, you may also specify subsystem-defined parameters.

The SUBSYS parameter can only be used if Subsystem Support (SU29) has been installed.

For additional information about the SUBSYS parameter and subsystem-defined parameters, refer to the documentation for the particular subsystem.

```
SUBSYS=(subsystem name [,parm1 [,parm2] ... [,parm254]])
```

subsystem name
specifies a one to four character subsystem name; the first character must be either alphabetic or national and the remaining characters must be either alphameric or national. A subsystem name must be specified. The system programming staff at your installation can identify the subsystems that are installed at your installation.

parm1,parm2,...parm254
subsystem defined parameters conforming to the following rules:

- A parameter can consist of alphameric, national, or special characters (see Figure 18 for a list of special characters). If you code special characters, enclose the entire parameter in apostrophes. If you want an apostrophe to be treated as part of a parameter, code two consecutive apostrophes.
- Each parameter and any enclosing apostrophes must start and end on the same card.
- If you omit a parameter that the subsystem considers positional, code a comma in its place.

Rules for Coding

- If only the subsystem name is specified, parentheses are not required.
- For a list of the parameters that are mutually exclusive with SUBSYS, see Figure 24. The specified subsystem may define additional mutually-exclusive parameters.
- If the following parameters are specified, they are syntax checked and then ignored by MVS:

COPIES	FCB	UNIT
DEST	OUTLIM	
DISP	SPACE	

- If DUMMY is specified with SUBSYS, the subsystem-defined parameters are syntax checked by the subsystem, and if they are acceptable, the request is assigned a dummy status by MVS.
- The specification of the SUBSYS parameter on overriding DD statements causes the UNIT parameter on the overridden DD statement to be ignored.
- If the DUMMY parameter is specified in a procedure, specifying the SUBSYS parameter on a DD statement that overrides a DD statement in the procedure nullifies the DUMMY specification.

Examples of the SUBSYS Parameter

```
//DD1    DD       DSN=ANYDS,DISP=OLD,SUBSYS=ABC
```

Subsystem ABC processes the data set ANYDS.

```
//DD1    DD       DSN=ANYDS,DISP=OLD,SUBSYS=(XYZ2,
                  'KEYWORD=DATA VALUE1')
```

Subsystem XYZ2 processes the data set ANYDS and the parameter. The parameter is enclosed in apostrophes because it contains an equal sign and a blank which are special characters.

```
//DD1    DD  DSN=ANYDS,DISP=OLD,SUBSYS=(XYZ2,IKJ2,
//             'NAME=''MODULE1''',DATE=10/21/77')
```

Subsystem XYZ2 processes the data set ANYDS and the parameters. The character string MODULE1, will be enclosed in apostrophes when MVS passes it to subsystem XYZ2.

The SYSOUT Parameter—*keyword, optional*

The SYSOUT parameter assigns an output class to an output data set.

For further information on the SYSOUT parameter, see "Obtaining Output" for either JES2 or JES3.

```
SYSOUT=(class-name [,program-name ][,form-name ] )
                   [,               ][,code-name ]
```

class-name
specifies, as an alphameric character (A-Z, 0-9) or *, the class associated with the output device to which you want your output data set written.

program-name
specifies the member name of an installation-written program in the system library that is to write the output data set, instead of JES2 or JES3. If a user-written writer is specified, it is executed under the control of an external writer rather than by JES2 or JES3. Notify the operator that such a data set exists so he will start an external writer. Two names, INTRDR and STDWTR are reserved for JES2 and JES3. (For their use and definition, see **OS/VS2 System Programming Library: Job Management.**)

form-name
is a 1-4 alphameric or national character string that specifies that the output data set should be printed or punched on a special output form.

code-name
(JES2) is a 1-4 alphameric or national character string that identifies the OUTPUT statement from which output characteristics will be obtained.

Rules for Coding

- Parentheses are not needed when only the class-name is specified.
- Refer to Figure 24 for parameters that are mutually exclusive with SYSOUT.
- To print the output data set and the messages from your job on the same output listing, specify the same output class in the SYSOUT parameter as you specified for messages in the MSGCLASS parameter. Or, specify SYSOUT=* for all data sets you want to default to the MSGCLASS output class. If you specify program-name, form-name, or code-name and do not specify class-name, the sysout class will default to the MSGCLASS output class.
- INTRDR causes the data set to be treated as a job stream.
- The specification of the SYSOUT parameter on overriding DD statements causes the UNIT parameter on the overridden DD statement to be ignored. If the DSNAME parameter was specified on the DD statement that was overridden, a warning message is issued.
- Do not make backward references to a DD statement for which SYSOUT was specified.
- When using JES2, the DCB=PRTSP=2 parameter is ignored when coded with SYSOUT.
- Do not code the program-name subparameter of the SYSOUT parameter when coding the userid subparameter of the DEST parameter. These subparameters are mutually exclusive. For example, you cannot code:

```
//DDAB      DD      SYSOUT=(D,program-name),DEST=(node,userid)
```

You can code:

```
//DDAC      DD      SYSOUT=D,DEST=(node,userid)
//DDAC      DD      SYSOUT=(D,program-name),DEST=node
```

Examples of the SYSOUT Parameter

```
//DD1       DD      SYSOUT=P
```

Specifies that the data set is to be written to the device corresponding to class P.

```
//JOB50     JOB     ,'C. BROWN',MSGCLASS=C
//STEP1     EXEC    PGM=SET
//DDX       DD      SYSOUT=C
```

The DD statement named DDX specifies that the data set is to be written to the device corresponding to class C. Since the classnames in the SYSOUT parameter and the MSGCLASS parameter on the JOB statement are the same, the system messages resulting from this job and the output data set can be written to the same unit record device.

```
//DD5       DD      SYSOUT=(F,,2PRT)
```

Specifies that the data set is to be written to the device corresponding to class F and the output data set is to be printed on a special form. The form name is 2PRT.

The TERM Parameter—*keyword, optional*

The TERM parameter notifies the operating system that a data set is coming from or going to a time-sharing terminal.

```
TERM=TS
```

TS
indicates to the system that the input or output data being defined is coming from or going to a time sharing terminal.

Rules for Coding

- Concatenate a DD statement with a DD statement that contains TERMS=TS only if it is the last DD statement in a job step.
- Code only the DCB and SYSOUT parameters with the TERM parameter. Any other parameters coded on a DD statement with TERM are ignored.
- If you code TERM on a SYSOUT DD statement and if the job is from a foreground terminal, the output goes to the terminal; if the job is submitted in batch, the output goes to whatever has been designated as the output device according to the definition given to SYSOUT.
- TERM=TS is only effective when specified in a Logon procedure.

Examples of the TERM Parameter

```
//DD1       DD      TERM=TS
```

This data set (DD1) is either coming from or going to a time-sharing terminal.

```
//DD3       DD      UNIT=2400,DISP=(MOD,PASS),TERM=TS,LABEL=(,NL),
//                  DCB=(LRECL=80,BLKSIZE=80)
```

All of the parameters in this example except TERM and DCB are ignored.

The UCS Parameter—*keyword, optional*

The UCS parameter describes the character set to be used for printing an output data set on a 1403 or 3211 printer or for SYSOUT.

For further information on the UCS parameter, see "Obtaining Output" for either JES2 or JES3 and **OS/VS2 System Programming Library: Data Management**.

```
UCS=(character set code[,FOLD][,VERIFY])
                       [,   ]
```


character set code
specifies 1-4 alphameric characters that identify the special character set you want for printing the data set. Figures 10 and 12 list the character set codes that are standard for JES2 and JES3.

FOLD
specifies that you want the chain or train corresponding to the desired character set loaded in the fold mode. The fold mode is described in the publication **IBM 2821 Control Unit**. The fold mode is most often requested when uppercase and lowercase data is to be printed only in uppercase.

VERIFY
specifies that the operator is to visually verify that the character set image corresponds to the graphics of the correct chain or train which is mounted before the data set is printed. The character set image is displayed on the printer before the data set is printed.

Default: For the 3211, the image currently in the buffer. For JES2, the buffer value must have a default flag. For JES3, by installation default or by job class.

Rules for Coding

- Parentheses are not needed when only the character set code is specified.
- If the chain or train mounted on the printer does not correspond to a valid character set, the operator is requested to identify the character set to be used, and mount the corresponding chain or train.
- If you code the UCS parameter and the data set is not written to a printer with the universal character set (UCS) feature, the UCS parameter is ignored.
- The FOLD and VERIFY subparameters are ignored for SYSOUT data sets.
- Code a comma when FOLD is omitted and the VERIFY subparameter follows.
- For both the 3211 and 1403 printers, you can code the UCS parameter with the UNIT parameter.
- Refer to Figure 24 for parameters that are mutually exclusive with UCS. UCS is also mutually exclusive with the DCB subparameters RKP and CYLOFL.
- In order to use a particular special character set, an image of the character set must be contained in SYS1.IMAGELIB and the chain or train corresponding to the character set must be available for use. 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)
```

Defines an output data set that is to be written to a 1403 printer. The UCS parameter requests that the data set be written using the chain or train corresponding to the special character set with the code YN. Since VERIFY is coded, the character set image will be displayed on the printer before the data set is printed.

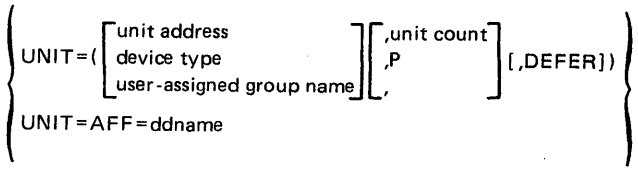
```
//DD2         DD        SYSOUT=G,UCS=PN
```

Defines an output data set that is to be written to the unit record device that corresponds to class G. If the device is a printer with the universal character set, the request in the UCS parameter for the special character set with the code PN will be recognized. Otherwise, the UCS parameter is ignored.

The UNIT Parameter—*keyword, optional*

The UNIT parameter specifies the types and number of devices you want assigned to a data set.

For further information on the use of the UNIT parameter, see "Requesting Units and Volumes".



unit address
specifies 3 hexadecimal characters that identify a particular unit by its address, which consists of the channel, control unit, and unit numbers.

device type (generic name)
is an IBM-supplied name (for example, 2314) that identifies a particular device by its device number. A list of IBM device types is included in **OS/VS2 System Programming Library: System Generation Reference**. Or, obtain the list from your system programmer.

user-assigned group name (esoteric name)
is a 1 to 8 alphameric character name that identifies a particular group of devices. The user-assigned name and the devices that make up a group are specified during system generation.

unit count
is a value from 1 to 59 that indicates the number of devices you want assigned to the data set.

P
indicates that the number of units to be allocated is equal to the number of volumes specified on the VOLUME parameter (the volume count subparameter or the number of serial numbers, whichever is higher).

DEFER
specifies that the system should assign a device(s) to the data set but the volume(s) on which the data set resides should not be mounted until the data set is opened.

AFF
indicates that within a job step, different data sets residing on different volumes can be allocated to the same unit provided that the volumes are removable (unit affinity).

ddname
is the name of an earlier DD statement in the job step that defines a data set with which you want unit affinity.

General Rules for Coding

- If you receive a passed data set or refer to a cataloged data set or earlier DD statement for volume and unit information (VOL=REF=reference), the system will assign one device, even if more devices were requested in an earlier DD statement. Therefore, you must code unit count for more than one device.
- Parentheses are not needed when only the first subparameter is coded.
- Code a comma when the unit count or P is omitted and the DEFER subparameter follows. One device is assigned to the data set.

- Do not identify a device by its address unless it is absolutely necessary. Specifying a unit address limits unit assignment and can result in a delay of the job if the unit is being used by another job.
- For installations where 3340 drives with and without the fixed head feature exist, the device type should not be used for the UNIT parameter. Instead, use the unit address or a user-assigned group name.
- If you code SYSOUT and UNIT on the same statement, the SYSOUT specification overrides the UNIT specification.
- You can receive more units than you specified if you have specified volume affinity and/or a permanently resident or reserved volume.
- Refer to Figure 24 for parameters that are mutually exclusive with UNIT.

Using the 3330 mod 11

If you request a 3330 mod 11, code UNIT=3330-1.

Using Mass Storage Volumes (MSS)

- If you are using mass storage volumes, code UNIT=3330V.
- To extend multivolume data sets to a non-mounted volume, the unit count must be less than the volume count.
- If an old, multivolume data set resides on volumes within a group, specify parallel mount or specify unit count equal to the number of volumes containing the data set.
- Deferred mounting should not be specified for volumes belonging to a MSVGP if there are new data set requests in that job step using MSVGP from the same group. The delay in selection can result in volume conflicts within the job or between jobs causing performance slowdown.

Rules for Coding User-assigned Group Names

- A user-assigned group name can identify a device or a group of devices. The group can consist of devices of the same type or class, direct access and tape device classes, or different types of devices.
- When you code a user-assigned group name, you allow the system to assign any available device from the group. If a group consists of only one device, the system will assign that device.
- If a group consists of more than one device type, the units requested are allocated from the same device type. For example, if SYSDA contains 3330 and 2314 device types, a request for two units would be allocated to two 3330s or to two 2314s.
- If a data set that was created using the user-assigned name subparameter is to be extended, additional units allocated to it will be of the same type as specified in the original group name. However, the units allocated to the data set may not necessarily be of that same name.

The unit count, volume count, and volume serial number may be used to determine the number of units and volumes required. The greatest of the three numbers is used.

Rules for Coding DEFER

- To defer mounting a volume, DEFER must be specified or implied for all DD statements that reference the volume.
- If you request deferred mounting of a volume and the data set on that volume is never opened by the processing program, the volume is not mounted during the execution of the job step.
- DEFER is ignored if a new, direct access data set is specified.

Rules for Coding AFF

- You can conserve the number of devices used in a job step by requesting that an existing data set be assigned the same device(s) as the data set defined on the DD statement specified by ddname.
- UNIT=AFF and DISP=NEW are mutually exclusive parameters if the referenced request is for a direct access device. If coded, the job terminates. If the referenced request is eligible for both tape and direct access devices, the job does not terminate; both requests are allocated to tape devices.
- If UNIT=APF is specified for a SYSIN (DD* or DD DATA or SYSOUT DD statement, the operating system ignores UNIT=AFF and defaults the unit name to SYSALLDA. SYSALLDA is the system-defined unit name for all direct access devices defined to the system.

Unit Override

Volume serial information is obtained from a volume reference to a data set name, a volume reference to an earlier DD statement (VOL=REF), a passed data set or a cataloged data set. The unit description is also available from these same sources. However, you can override the retrieved unit information if the unit you specify is a subset of the retrieved unit. For example, if the retrieved unit grouping is 2314, and the specified unit description is 2314A (a subset of 2314) or device address 237 (a subset of 2314), then the only units considered for allocation are those contained within 2314A or at the device address of 237.

Examples of the UNIT Parameter

```
//STEP2     EXEC    PGM=POINT
//DDX       DD      DSNAME=EST,DISP=MOD,VOLUME=SER=(42569,42570),
//                  UNIT=(2314,2)
//DDY       DD      DSNAME=ERAS,DISP=OLD,UNIT=2400-2
//DDZ       DD      DSNAME=RECK,DISP=OLD,
//                  VOLUME=SER=(40653,13262),UNIT=AFF=DDX
```

The DD statement named DDZ requests that the system assign the same unit to this data set that it assigns to the data set defined on the statement named DDX. Since DDX requests two devices, the same two devices are assigned to the data set defined on DDZ.

```
//DD1       DD      DSNAME=AAG3,DISP=(,KEEP),
//                  VOLUME=SER=13230,UNIT=2400
```

Defines a new data set and requests that the system assign any 2400 9-track (that can read/write 800 bpi) tape drive to the data set.

```
//DD2       DD      DSNAME=X.Y.Z,DISP=OLD,UNIT=(,2)
```

Defines a cataloged data set and requests that the system assign two devices to the data set. The device type will be obtained from the catalog.

```
//DD3       DD      DSNAME=COLLECT,DISP=OLD,
//                  VOLUME=SER=1095,UNIT=(3330,,DEFER)
```

Defines an existing data set that resides on a direct access volume and requests that the system assign a 3330. Since DEFER is coded, the volume will not be mounted until the data set is opened.

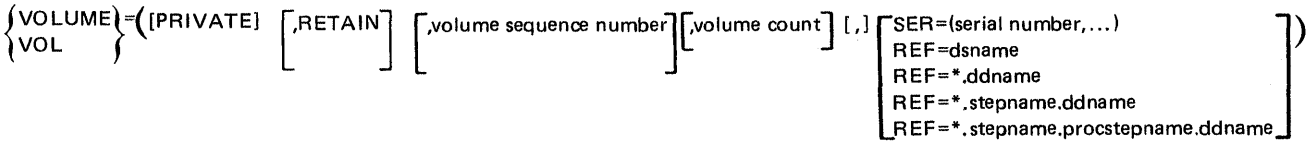
```
//STEPA     DD      DSN=FALL,DISP=OLD,UNIT=237
```

The volume and unit device type are retrieved from the catalog. You can override the unit by specifying UNIT=237 if that unit is a subset of the device type specified in the catalog.

The VOLUME Parameter—*keyword, optional*

The VOLUME parameter identifies the volume(s) on which a data set resides or will reside.

For further information on the use of the VOLUME parameter, see "Requesting Units and Volumes."



PRIVATE
specifies that:

- no output data set is to be allocated to this volume unless the volume is specifically requested.
- the volume is to be demounted after its last use in the job step, unless RETAIN is coded or the data set is passed.

RETAIN
For a private tape volume, RETAIN specifies that this volume is not to be demounted and will not be rewound after its last use in the job step, or at the end of the step. For a public tape volume, RETAIN specifies that this volume is to be retained at the system if it was demounted during the job.

volume sequence number
specifies which volume of an existing multivolume data set you want used to begin processing.

volume count
specifies the maximum number of volumes an output data set requires.

,
specifies that either the SER or REF subparameter follows and one or more subparameters precede it.

SER=
indicates that serial numbers of the volumes on which the data set resides or is to reside, are specified.

(serial number,...)
specifies the serial numbers of the volumes on which the data set resides or will reside.

REF=
indicates that the serial numbers of the volumes on which the data set resides or is to reside are identified on an earlier DD statement in the job or in the catalog or an earlier passed data set.

dsname
specifies the name of a cataloged or passed data set. The system locates the information about the data set and assigns your data set to the same volumes as are assigned to the cataloged or passed data set. The data set name specified can not be a generation data group (GDG) name.

***.ddname**
specifies that the system must obtain the volume serial numbers from an earlier DD statement named "ddname" in the same job step.

***.stepname.ddname**
specifies that the system must obtain the volume serial numbers from a DD statement named "ddname", which was defined in an earlier job step name "stepname."

***.stepname.procstepname.ddname**
specifies that the system must obtain the volume serial numbers from a DD statement named "ddname", which was defined in an earlier procedure step named "procstepname"; the procedure step is part of a procedure that was called by an earlier job step named "stepname."

General Rule for Coding

Refer to Figure 24 for parameters that are mutually exclusive with VOLUME.

Rules for Coding when using Mass Storage Volumes (new data sets only)

- VOL=SER is mutually exclusive with the MSVGP parameter.
- The SPACE parameter is required when VOL=SER is coded.
- To guarantee allocation to SYSGROUP for a nonspecific request, specify VOL=PRIVATE or MSVGP=SYSGROUP.

Rule for Coding PRIVATE

Parentheses are not needed when only PRIVATE is coded.

Rules for Coding RETAIN

- Code a comma when RETAIN is omitted and the volume sequence number or the volume count subparameter follows.
- Coding RETAIN on a direct access DD statement has no effect on volume handling.
- Coding RETAIN does not ensure that the volume cannot be unloaded by the operator or allocated and demounted by another job step. This can occur when the job step that specified RETAIN is not allocated to the unit on which the volume is mounted.
- Coding RETAIN on a DD statement that is set up by JES3 has no effect since the JES3 main drive scheduler (MDS) ignores RETAIN.

Rules for Coding Volume Sequence Number

- The volume sequence number must be less than or equal to the number of volumes on which the data set exists; it can be from 1 to 255. If a unit count greater than the remaining specific volumes is specified, nonspecific volumes are assigned to the remaining units.
- Normally, you code a volume sequence number when you have not specified volume serial numbers on the DD statement (that is, you are retrieving a cataloged data set or you have coded a reference to an earlier DD statement or data set). If you code both a volume sequence number and a volume serial number in the VOLUME parameter, the system begins processing with the volume that corresponds to the volume sequence number.
- The volume sequence number must correspond to a specific volume serial number or the job fails.
- The volume sequence number is ignored for NEW data sets.
- The volume sequence number overrides a DISP=MOD parameter.
- Code a comma when the volume sequence number is omitted and the volume count subparameter follows.

Rules for Coding Volume Count

- The volume count value can range from 1 to 255.
- If a volume count of 1 to 5 is specified, the maximum number allowed is 5; if a count of 6 to 20 is specified, the maximum number allowed is 20; if a volume count greater than 15 is specified, the maximum number allowed will be 5 plus a multiple of 15 up to a maximum of 255. The total volume count for all DD statements in one job step cannot exceed 4095.
- If the volume count is greater than the number of specific volume serial numbers, non-specific volumes are added to make up the total. If the number of specific volume serial numbers is greater than the volume count, the volume count is ignored.
- If the request is for a nonspecific, non-private volume request for a direct access device, the volume count is ignored. (The number of volumes equals the unit count value coded on the UNIT parameter.)
- When you make a specific volume request and the data set might require more volumes than there are serial numbers, specify in the volume count subparameter the total number of volumes that might be used. By requesting multiple volumes in the volume count subparameter, you can ensure that the data set can be written on more than one volume if it exceeds one volume.
- A nonspecific, private volume request is treated like a specific request if more than one volume is needed; that is, the volume count subparameter determines the number of volumes allocated.

Rules for Coding SER

- You can specify a maximum of 255 volume serial numbers for each DD statement.
- You should not specify duplicate volume serial numbers on the SER parameter. (If duplicate volume serial numbers are specified on the SER subparameter, the results vary depending on the job entry subsystem in use.)
- Volume serial information is obtained from a volume reference to a data set name, a volume reference to an earlier DD statement (VOL=REF), a passed data set, or a cataloged data set.
- A volume serial number must be 1 to 6 characters in length. If the number is less than 6 characters, it will be padded with trailing blanks. It can contain any alphameric and national characters, and the hyphen. You must enclose any volume serial number than includes special characters other than the hyphen in apostrophes whenever you code that number in the volume parameter.
- When using some typewriter heads or printer chains, difficulties in volume serial recognition may arise if you use other than alphameric characters.
- The SER subparameter appears as the last subparameter in the VOLUME parameter. Follow SER= with the volume serial numbers. The serial numbers must be enclosed in parentheses unless there is only one serial number. If SER is the only subparameter you are coding, you can code VOLUME=SER=(serial number,...) or VOLUME=SER= serial number.
- If you code VOL=SER=, and do not code a serial number, the system treats your request as a non-specific volume request and does not issue an error message.
- Do not use SCRTCH, PRIVAT, or Lnnnnn L with five numerics) as a volume serial number because they are used as special messages to notify the operator to mount a volume. For optical readers, if no volume serial number is specified, VOLUME=SER=OCRINP is assumed.
- Each volume must have different volume serial numbers regardless of the volume type (for example, tape and disk).

Rules for Coding REF

- To refer the system to a cataloged data set or to a data set passed earlier in the job that has not been assigned a temporary data set name, code REF as the last subparameter in the VOLUME parameter. Follow REF with the data set name of the cataloged or passed data set. The data set name specified cannot contain special characters except for periods used in a

qualified name unless the name is enclosed in apostrophes. And the data set name specified must not be a generation data group (GDG) name.

- If a volume reference is made to a SYSIN (DD* or DD DATA) or SYSOUT DD statement, the operating system ignores the reference and defaults the unit name to SYSALLDA. SYSALLDA is the system-defined unit name for all direct access devices defined to the system.
- To refer the system to a data set defined earlier in the job that was not passed or was passed but assigned a temporary name, code REF with a backward reference to the DD statement that contains the volume serial numbers.
- A volume reference to a new nonspecific multivolume tape data set that was created in a prior step is considered a specific volume request for all subsequent steps. As such, only the last volume used when the data set was created will be assigned to all subsequent DD statements that specify the REF subparameter. If the data set is extended to further volumes, this new volume information is not available to following DD statements.
- If the ddname refers to a DD statement that defines a dummy data set, it also is assigned a dummy status.
- Do not make a volume reference (VOL=REF=) to a DD statement from which you are allocating space by using the SUBALLOC keyword.

  The system automatically assigns volume affinity to the DD statement with the original space request.
- When referring the system to a data set that resides on more than one tape volume, the system begins with the last volume. When you refer the system to a data set that resides on more than one direct access volume, the system assigns all of the volumes. In either case, you can code the volume count subparameter if additional volumes may be required.
- When coding a volume reference to a previous DD statement that uses user-assigned names, the system will allocate from the same device type name you made reference to rather than from user-assigned group names.
- When referring to a multi-volume VSAM data set, you will receive only the first device type.
- If the reference is to a DD statement, the label type is also copied from the referenced DD.

Checkpoint/Restart

- When a checkpoint data set is not cataloged, code the VOLUME parameter and specify the volume serial number of the volume on which the checkpoint entry is written.
- If a checkpoint data set is cataloged, you do not need to code the VOLUME parameter unless the checkpoint entry exists on a tape volume other than the first volume of the data set; then, code either a volume sequence number or the volume serial number. If you code the volume serial number, you must code the UNIT parameter.

Examples of the VOLUME Parameter

```
//DD1       DD      DSNAME=STEP,UNIT=2314,DISP=OLD,
//                  VOLUME=(PRIVATE,SER=548863)
```

Defines an existing data set and informs the system that the data set resides on the volume whose serial number is 548863. Since PRIVATE is coded in the VOLUME parameter, the system will not assign the volume to any data set for which a nonspecific volume request is made and will cause the volume to be demounted at the end of the job.

```
//DD2       DD      DSNAME=QUET,DISP=(MOD,KEEP),UNIT=(2400,2),
//                  VOLUME=(,,,4,SER=(96341,96342))
```

Defines an existing data set that resides on the volumes whose serial numbers are 96341 and 96342, and requests that 4 volumes be used to process the data set if required. If more space is required, the operating system can assign a fifth volume.

```
//DD3          DD        DSNAME=QOUT,DISP=NEW,UNIT=2400
```

Defines a temporary data set and, by omission of the VOLUME parameter, requests that the system assign a suitable volume to the data set.

The Command Statement

Control Statement

The command statement specifies an operator command to be executed.

Note: All command statements are ignored by JES3.

For further information on commands and for descriptions of their operands, see **Operator's Library: OS/VS2 MVS System Commands.**

```
// command operand comments
```

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

The following JCL commands can be entered through the input stream.

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

Rules for Coding

- Follow the // in columns 1 and 2, with one or more blanks.
- Code the command (valid abbreviations are permitted).
- Follow the command with one or more blanks.
- Code any required operands. Separate each operand with a comma.
- Follow the operands with one or more blanks.
- Code any comments.
- The command statement cannot be continued.
- A command statement can appear immediately before a JOB statement, an EXEC statement, a null statement, or another command statement, but not before the first job in the input stream in JES2.
- If a command statement appears in the input stream between the boundaries of two jobs and it contains errors, the command will not be executed. Furthermore, you will receive no indication that the command was not executed.
- If you include a command statement as part of your job control statements, the command will usually be executed as soon as it is read. Because of this, it is not likely that the command will be synchronized with the execution of the job step to which it pertains. Therefore, you should preferably tell the operator which commands you want issued and when they should be issued, and let him issue them.
- Disposition is determined by JES2 according to installation options specified for each job class.

Example of the Command Statement

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

Displays the number and user-id of all active time-sharing users of the system.

Control Statement

The comment statement specifies a comment to be included in the output listing.

```
//*comments
```

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

Rules for Coding

- Code the comments in columns 4 through 80.
- You cannot continue comment statements using continuation conventions. If you cannot include all of the comments on one comment statement, code another comment statement.
- The comment statement can appear anywhere after the JOB statement, including between continuations of statements.
- With the MSGLEVEL parameter, you can request an output listing of all the control statements processed in your job. You will be able to identify comment statements by the appearance of *** in columns 1, 2, and 3.

Example of the Comment Statement

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

Control Statement

The delimiter statement indicates the end of data submitted through an input stream for a step.

```
/*  comments
```

The delimiter statement consists of the characters /* in columns 1 and 2 and the comments field. You must have at least one blank before the comments field.

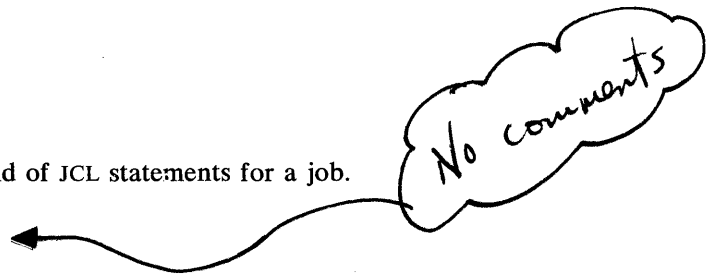
Rules for Coding

- The system recognizes a delimiter other than /* if the DLM parameter is coded on the DD statement defining the data.
- Code /* (or the value assigned in the DLM parameter) in columns 1 and 2, followed by any comments you have. The comments cannot be continued.
- The beginning of data to be submitted through an input stream is indicated by a DD * or DD DATA statement.
- If the data is preceded by a DD * statement and you do not code the DLM parameter, you need not code a delimiter statement.

Example of the Delimiter Statement

```
//JOB54     JOB ,'C BROWN',MSGLEVEL=(2,0)
//STEPA     EXEC PGM=SERS
//DD1       DD *
            .
            .
            .
            data
            .
            .
            .
/* END OF DATA FOR THIS STEP
```

The Null Statement



Control Statement

The null statement indicates the end of JCL statements for a job.

```
//
```

The null statement consists only of the characters // in columns 1 and 2. The remainder of the statement must be blank.

Rules for Coding

- The null statement is ignored by JES2.
- Place a null statement at the end of a job's control statements or at the end of all the statements in an input stream.
- If you do not follow the job's control statements and data with a null statement, the system places the job on the queue when it encounters another JOB statement in the input stream.
- If the job is the last job in the input stream and it is not followed by a null statement, the system recognizes it as the last job in the input stream and places it on the queue.
- The system flushes statements between a null statement and the next valid JOB statement.
- If a null statement follows a control statement that is being continued, the system treats the null statement as a blank comment field and assumes that the control statement contains no other operands.

Example of the Null Statement

```
//MYJB      JOB     ,'C BROWN'
//STEP1     EXEC    PROC=FIELD
//STEP2     EXEC    PGM=XTRA
//DD1       DD      UNIT=2400
//DD2       DD      *
                    .
                    .
                    .
                    data
                    .
                    .
                    .
/*
//
```

Control Statement

The PEND statement marks the end of an in-stream procedure.

```
//name  PEND    comments
```

The PEND statement consists of the characters // in columns 1 and 2 and three fields — the name field, the operation (PEND) field, and the comments field.

Rules for Coding

- Code // in columns 1 and 2 then code a name (1 to 8 characters) or one or more blanks.
- If you code a name, follow it with one or more blanks.
- Code PEND, and follow it with one or more blanks.
- Code any desired comments.
- Do not continue a PEND statement. The PEND statement terminates an in-stream procedure at that point, whether or not the statement is continued. The PEND statement must not be included in cataloged procedures.

Examples of the PEND Statement

```
//PROCEND1 PEND THIS STATEMENT IS REQUIRED FOR INSTREAM PROCEDURES
```

This PEND statement contains a comment.

```
// PEND
```

A PEND statement can contain only the coded operation field preceded by // and one or more blanks and followed by blanks.

Control Statement

The PROC statement is the first control statement in an in-stream procedure; the PROC statement can also be the first control statement in a cataloged procedure. In either an in-stream procedure or a cataloged procedure, a PROC statement can be used to assign default values to symbolic parameters in the procedure.

```
//name      PROC    operands     comments
```

The PROC statement consists of the characters // in columns 1 and 2 and four fields — the name field, the operation (PROC) field, the operand field, and the comments field.

Rules for Coding

- A PROC statement is required for an in-stream procedure; it must appear as the first control statement of the in-stream procedure.
- A PROC statement is optional for a cataloged procedure; if a PROC statement is included in a cataloged procedure, it must appear as the first control statement.
- Code // in columns 1 and 2; then code a 1 to 8 character name or one or more blanks. A name is required for in-stream procedures.
- Cataloged procedures with no symbolic parameters can be created and executed.
- If you code a name, follow it with one or more blanks. Then code PROC, followed by one or more blanks.
- In the operand field, you can assign default values to symbolic parameters in a procedure. Code a comma after a symbolic parameter and its default value, if you are coding more than one. Do not code a comma after the last symbolic parameter and its default value.
- The operand field is required in an in-stream procedure only if symbolic parameters are defined as in the example at the end of this section.
- Follow the operands with one or more blanks and any desired comments.
- You can continue the PROC statement onto another statement. Code // in columns 1 and 2 of the continuation statement.
- To assign a value to a symbolic parameter, code:
  symbolic parameter=value
  Omit the ampersand that precedes the symbolic parameter in the procedure.
- The value assigned to a symbolic parameter can be any length, but it cannot be continued onto another statement.
- If the symbolic parameter value contains special characters, enclose the value in apostrophes (the enclosing apostrophes will not be considered part of the value).
  If the special characters include apostrophes, you must code each apostrophe as two consecutive apostrophes.
- If you assign more than one value to a symbolic parameter with some other information, (for example, &JOBNO.321), the information and value cannot exceed a total of 120 characters.
- You can override a default value appearing on a PROC statement by assigning a value to the same symbolic parameter on the EXEC statement that calls the procedure.

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=2400,
//                  VOLUME=SER=888888
//DD2       DD      DSNAME=&LIBRARY,DISP=(OLD,KEEP),UNIT=2314,
//                  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 will be used when the procedure is called and values have not been assigned to the symbolic parameters by the programmer.

```
//CARDS     PROC
```

This PROC statement can be used to mark the beginning of an in-stream procedure named CARDS.

Coding JES2 Control Statements

Code JES2 control statements with JCL statements to control the input and output processing of jobs. Rules for coding JCL, including syntax, in the section, "Coding JCL Statements," apply to the JES2 control statements. However, there are additional rules for coding JES2 statements. They are:

- Columns 1 and 2 always contain the characters /*.
- JES2 statements other than the OUTPUT statement cannot be continued. You can use multiple control statements if more than one statement is needed.
- Do not place JES2 control statements in a cataloged procedure; they are ignored.
- If you code more than one statement with the same parameters, the last statement coded will override any other statements.
- If you code more than one of the same parameters on the same statement, the last parameter coded will override any other parameters.
- You can code the JES2 control statements in any order. However, the COMMAND and the PRIORITY statements must be placed in front of the JOB statement and all other JES2 statements should follow the JOB statement.
- The JOBPARM statement overrides the installation default but can itself be overridden by a specific output statement.

Note: The following JES2 statements are described in **System Programming Library: JES2 Installation, Initialization, and Tuning.**

```
/*DEL
/*EOF
/*PURGE
/*SCAN
```

The Command Statement

Control Statement

The command statement specifies JES2 operator commands that can be entered through the card reader or the system console. Examples in this book illustrate the format for commands entered through the card reader. Commands entered through the system console should omit the /* from the message.

For a detailed description of the command statement and the names of the correct JES2 verbs and operands, see **Operator's Library: JES2 Commands.** The command statement consists of the characters /* in columns 1 and 2. Column 3 contains a character that is established at JES2 initialization by the installation or defaults to '$'. There are two fields — a JES2 command verb starting in column 4 followed by one or more operands. An "N" may be coded in column 72. Columns 73-80 are ignored.

```
/*$command verb  operand [,operand...][N]
```

command verb
an operand indicating which JES2 operator command is to be performed.
The following JES2 commands can be entered through the input stream.

$A	$E	$I	$O	$T
$B	$F	$L	$P	$TR
$C	$G	$M	$R	$VS
$D	$H	$N	$S	$Z

operand
one or more variable length operands.

N
indicates that the command will not be repeated on the operator's console.

Rules for Coding

- Code as many command statements as are needed, but do not continue them from one statement to the next.
- Command statements must be placed before jobs being entered through the input stream. Any command statements within a job will be ignored.
- Commands that are entered on the command statement are executed immediately. They cannot be linked with any execution process of a job.
- JES2 commands entered through the input stream are of the form /*$command. The $ is a JES2 initialization option.
- Commands following the /*XMIT statement are processed and displayed at the execution node. All other commands entered through the card reader are processed and displayed at the input node.

Example of the Command Statement

```
/*$SI3-5
```

Starts initiators three through five. The command is $S and the operand is I3-5.

The JOBPARM Statement

Control Statement

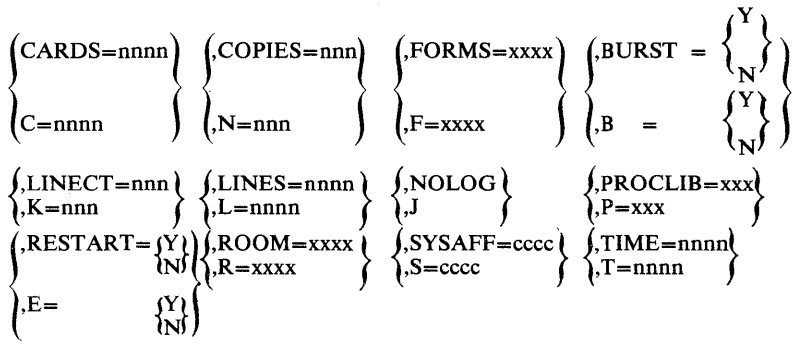
The JOBPARM statement specifies job related parameters for JES2.

The JOBPARM statement consists of the characters /* in columns 1 and 2, the word JOBPARM in columns 3-9, a blank in column 10, and parameters in columns 11-71. Columns 72-80 are ignored.

For further information, see "Obtaining Output (JES2)."

```
/*JOBPARM parameters
```

Code one or more of the following parameters in the longer form (full word) or the shorter form (one letter abbreviation).



CARDS=nnnn
a value estimating the number of output cards from this job (from 0 to 9999999 cards).

COPIES=nnn
a value indicating the number of printed output copies of a job related output that is to be produced (from 1 to 255 copies). The upper limit of this value can be reduced during JES2 initialization.

FORMS=xxxx
an alphameric value indicating the print and punch forms for this job's output that are not further defined in this job (from 1 to 4 characters).

BURST=Y or N
indicates the default burst characteristic of all SYSOUT data sets produced by this job. This characteristic is used to either burst the data set (Y), or to continuously fold the data set (N), when the data set is directed to an IBM 3800 printer with the burst option installed.

LINECT=nnn
a value showing the number of lines to put on each output page for JES2 page overflow processing (from 0 to 255 lines).

LINES=nnnn
a value estimating the number of output lines from this job — in thousands of lines (from 0 to 9999).

NOLOG
a parameter meaning that you do not want the JES2 job log as output. (The job log contains a list of job related console messages and operator replies produced during processing of the job.)

PROCLIB=xxx
an alphameric value indicating the DDNAME of the cataloged procedure library that is to be used to convert the JCL for this job. (This name refers to a DD statement in the JES2 cataloged procedure.)

RESTART
if this job is executing before a re-IPL and JES2 warm start and cannot be restarted from a step or checkpoint, JES2 does one of the following:

- Y indicates that the job is queued for re-execution from the beginning of the job.
- N indicates that no special action is to be taken.

If RESTART is not specified, N is assumed. However, the installation may override this default by JES2 initialization parameters.

ROOM=xxxx
an alphameric value indicating a programmer's room number to be placed on the job's separators for routing SYSOUT data sets back to the programmer (from 1 to 4 characters).

SYSAFF=cccc
1 to 7 system affinities can be specified indicating systems to be eligible to process this job. In order to specify more than one system, code: SYSAFF=(cccc,cccc,...). cccc is an alphameric value indicating one or more of the following:

- * indicates the system into which the job was read.
- ANY indicates any system in the JES2 multi-access spool configuration.
- cccc indicates a specific system. "cccc" must be the four alphameric character system-id of one of the systems in the JES2 multi-access spool configuration.
- ,IND when included after any of the above specifications, indicates systems scheduling in independent mode.

TIME=nnnn
a value estimating the job execution time in minutes of real time (from 0 to 279,620 minutes).

Rules for Coding

- Any JOBPARM statement parameter value will supersede the equivalent parameter value from the accounting field (in HASP format) of the JOB statement or from any preceding JOBPARM statement in this job's JCL. All of these statements override the default established by the installation.
- Any number of the above parameters may be placed on a single JOBPARM statement and as many JOBPARM statements as desired may be placed together with a given input stream. The JOBPARM statement cannot be continued.
- Place the JOBPARM statement after the JOB statement.
- If you code the PROCLIB parameter on the JOBPARM statement, the name of the DD statement should be in the JES2 cataloged procedure. If it is not, the JES2 default procedure is used.
- If you code LINECT=0, JES2 will not eject to a new page when the number of lines has exceeded the page limit that was established at JES2 initialization.
- The COPIES parameter does not apply when FREE= is specified on the SYSOUT DD statement.
- If you code COPIES=000, JES2 will default the copy count to one.
- The JOBPARM statements will be ignored on any node other than the node of execution. However, if the JOBPARM card is placed before the ROUTE card, it will be processed at both the input node and the execution node.

Example of the JOBPARM Statement

```
/*JOBPARM    L=60,R=4222,T=50
```

The three specifications mean the following:

L=60	The job's estimated printed output will be 60,000 lines.
R=4222	The programmer's room is 4222. This information will be placed in the separators for both printed and punched data sets.
T=50	The job's estimated execution time is 50 minutes.

The MESSAGE Statement

Control Statement

The MESSAGE statement permits you to send messages to the operator (via the operator console) at JES2 job input time.

The MESSAGE statement consists of the characters /* in columns 1 and 2, the word MESSAGE in columns 3-9, and at least one blank followed by the message. Columns 72-80 are ignored.

```
/*MESSAGE   message to be written
```

Rules for Placement

- Place the MESSAGE statement after the JOB statement to allow the job number to be appended to the beginning of the message.
- If the MESSAGE statement is not included within the boundaries of a job, the input device name is appended to the beginning of the message.
- The message will only be sent to the operator on the execution node unless the MESSAGE card is placed before the ROUTE card. The latter will cause the message to be printed on both the input and execution nodes.

Example of the MESSAGE Statement

```
/*MESSAGE CALL DEPT58 WHEN PAYROLL JOB IS FINISHED--EX.1946
```

Requests that the operator call department 58 when the payroll job is complete.

The NETACCT Statement

The NETACCT statement specifies an account number which is available to all the nodes in a network; the account number may be used as is or may be translated to local account numbers as desired. 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. Columns 72-80 are ignored.

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

network-account-number
From 1 to 8 alphameric characters representing this job's accounting through the network.

Rules for Coding

Observe the following rules for coding the NETACCT statement:

- Place the NETACCT statement after the JOB statement.
- If more than one NETACCT card is included, the last network account number is used.
- The NETACCT card is ignored on any node other than the input node.
- If a NETACCT statement is not supplied, JES2 attempts to find a network account number using the local account number in a table lookup.
- If both NETACCT and a local account number are supplied, the local account number is used on the input node.

Example of the NETACCT Statement

```
/*NETACCT NETNUM10
```

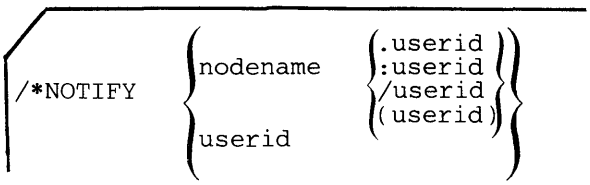
The network account number, NETNUM10, will be transmitted with the job to various nodes, available for local interpretation at each node.

The NOTIFY Statement

Control Statement

The NOTIFY statement causes a job's notification messages to be directed to the userid specified on the NOTIFY statement. Print, punch, and execution routings are not affected. Use of the NOTIFY statement overrides any NOTIFY specification on the JOB statement.

The NOTIFY statement consists of the characters /* in columns 1 and 2, the word NOTIFY in columns 3-8, and one or more blanks followed by a parameter. Columns 72-80 are ignored.



nodename.userid
a node name and user identifier (each from 1 to 8 characters) separated by a delimiter. Nodename must consist of alphameric characters. userid may contain any valid characters except the comma (,), right parenthesis ()), or blank character. Specification of nodename.userid causes the origin node field in the job's network job header to be set to the node specified on the /*NOTIFY statement even though the actual job origin node may be different.

userid
a user identifier of 1 to 8 alphameric characters. Specification of userid causes notification messages to be sent to the job's origin node. (The origin node name is not overridden by the /*NOTIFY statement.)

Examples of the NOTIFY Statement

```
/*NOTIFY VMNODE.VMUSER
```

Notification messages are sent to user VMUSER on node VMNODE.

```
/*NOTIFY TSOUSER
```

Notification messages are sent to user TSOUSER on the job's origin node.

The OUTPUT Statement

Control Statement

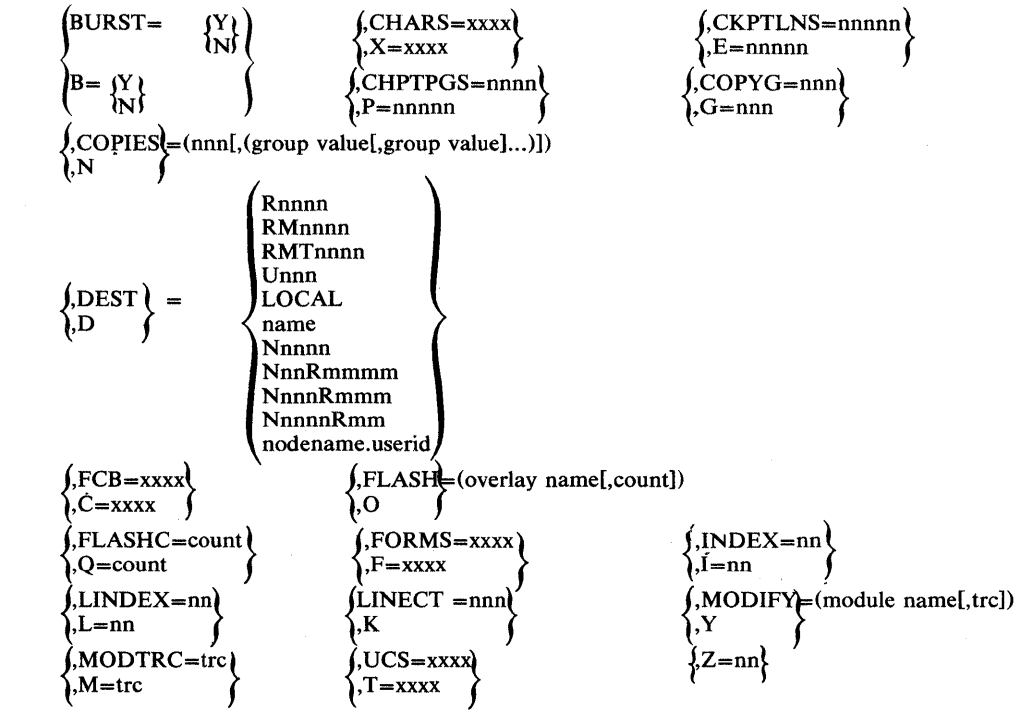
The OUTPUT statement specifies characteristics and/or options of a specific SYSOUT data set or group of SYSOUT data sets.

For further information on the OUTPUT statement, see "Obtaining Output (JES2)".

The OUTPUT statement consists of the characters /* in columns 1 and 2, the word OUTPUT in columns 3-8, and a code beginning in column 10 followed by a blank and the keyword parameters. Columns 72-80 are ignored.

```
/*OUTPUT    code    parameters
```

Code one or more of the following parameters in the longer form (full word) or the shorter form (one letter abbreviation).



code
alphameric characters referring to all SYSOUT data sets within your job whose code in the form number subparameter of the SYSOUT parameter matches the "code" specified on the OUTPUT statement (from 1 to 4 characters). Specifying code as "*" indicates that this OUTPUT statement is a continuation of the previous OUTPUT statement.
If the value you specify for code on the /*OUTPUT statement is the same as the value specified for the JES2 initialization parameter & SYSJCL, the values specified on this /*OUTPUT statement apply to the JES2-maintained system data sets.

BURST
Y indicates that the printed output from a 3800 printer is to be burst into separate sheets. N indicates that the printed output is to be in continuous, fanfold mode.

CHARS=xxxx
the name of a character arrangement table for a 3800 printer. Each name is 1 to 4 alphameric or national characters; from one to four names can be coded. To specify more than one name, code: CHARS=(xxxx,xxxx...).

CKPTLNS=nnnnn
a value (from 0 to 32767 for printers and 1 to 32767 for punches) that specifies the maximum number of lines or cards contained in a logical page. The default is the value specified in the JES2 initialization parameter for the device.

CKPTPGS=nnnnn
a value from 1 to 32767 that specifies the number of logical pages to be printed before the next checkpoint is taken. The default is the value specified in the JES2 initialization parameter for the device.

COPIES=(nnn[,(group value[,group value]...)])
copies specifies the number of copies of the data set to be printed, and if printing is done on a 3800, optionally specifies how the copies are to be grouped.

nnn
specifies the total number of copies of the data set to be printed, subject to an installation limit. nnn is ignored for the 3800 if group values are specified.

group value
describes the grouping of the printed copies for the 3800 printer only. Each group value specifies the number of copies of each page that are to be grouped together. (For example, a group value of 3 causes the first page of a data set to be printed three times before printing is started for the second page.) When group values are coded they override nnn. The total number of copies printed equals the sum total of the group values.

Default for nnn: 1
If you omit or incorrectly code the nnn parameter of COPIES, it defaults to 1 and a warning message is issued.

COPYG=nnn
a value that specifies how many copies of each page of the printed output are to be grouped together. The total number of copies printed equals the sum of the specified group values. The sum of the group values cannot be greater than 255. To specify more than one group value, code: COPYG=(nnn,nnn...). The maximum number of groups that can be specified is 8.

Note: This parameter applies only to the IBM 3800 Printing Subsystem (a nonimpact printer). If COPYG is coded and an impact printer is used, COPYG is ignored.

DEST=destination
One to four different destinations can be specified for each output data set. To specify more than one destination, code: DEST=(n[,n][,n][,n] n is an alphameric value indicating one of the following:

- Rnnnn, RMnnnn, or RMTnnnn - remote terminal, nnnn indicating a 1- to 4-digit numeric value (1 to 1000) specifying the remote terminal number.

Note: R0 is equivalent to LOCAL.

- Unnn - local terminal, nnn indicating a 1to 3-digit decimal value (1 to 255) specifying the local device with special routing.
- LOCAL - any local device.
- name - a 1- to 8-character (alphameric or national) name of a remote or local device (as defined by the system programmer).
- Nnnnn - a node identified by nnnn which is a 1-4 digit decimal number between 1 and 1000 inclusive.
- NnnRmmmm, NnnnRmmm, or NnnnnRmm where N designates a node and designates a remote work station that is connected to the node. nn, nnn, or nnnn is 1-99, 1-999, or

1-1000 and mm, mmm, or mmmm is 1-99, 1-999, or 1-1000. The maximum number of digits (n and m combined) can not exceed six. A specification of R0 is equivalent to the specification LOCAL at node N.

- nodename.userid - a node name and user or remote identifier (each from 1 to 8 characters) separated by a delimiter. The delimiter can be a period (.) as shown, a colon (:), a slash (/), or paired parentheses (enclosing the userid). Nodename must consist of alphameric characters. Userid can contain any valid characters except the comma (,), right parenthesis ()), R, RM, RMT, or blank character. This parameter is used to route information between JES2 nodes and eligible virtual machines at VM nodes or eligible JES3 nodes.

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

FCB=xxxx

an alphameric value indicating the data set forms control or carriage specifications (from 1 to 4 characters).

FLASH=(overlay name[,count])

identifies the forms overlay to be used on the 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 (1 to 4 alphameric or national characters). There is no system verification that the operator inserted the correct frame.

count

indicates the number of copies (from 1 to 1000) to be flashed with the overlay, beginning with the first copy printed.

Default for FLASH

If you omit this parameter and you did not specifiy FLASH on the DD statement or FLASHC on the OUTPUT statement. JES2 uses the default specified at JES2 initialization.

Default for count: 1000

If you specify 0 for count, JES2 uses the default.

FLASHC=count

a value, between 0 and 255, that indicates the number of copies to be flashed with the overlay, beginning with the first copy printed. For the 3800 printer, if FLASH is specified and FLASHC is omitted, all copies are flashed.

Note: The value you specify in the count subparameter of the FLASH parameter overrides any value you specify with the FLASHC parameter.

FORMS=xxxx

an alphameric value indicating the print and punch forms (from 1 to 4 characters).

INDEX=nn
a value indicating the data set indexing print position offset (to the right) for the 3211 printer (from 1 to 31).

LINECT=nnn
a value from 0 to 255 that specifies the number of lines that is printed on each page for JES2 overflow processing.

When you code LINECT on the /*OUTPUT statement it overrides the LINECT value on the /*JOBPARM statement and the linect value in the ACCOUNTING field parameter of the JCL JOB statement.

LINDEX=nn
a value indicating the data set indexing print position offset (to the left) for the 3211 printer (from 1 to 31).

MODIFY=(module name [,trc])
specifies the name of a copy modification module to be loaded into the 3800 Printing Subsystem. This module contains predefined data such as legends, column headings, or blanks, and specifies where and on which copies the data is to be printed. using the IEBIMAGE utility program.

module name
identifies a copy modification module used to replace blanks or data in the printed data set. module name is 1 to 4 alphameric or national characters.

trc
indicates the table reference character (0 to 3) that corresponds to a character arrangement table specified with the CHARS JCL parameter. The 3800 Printing Subsystem uses this table for printing of the copy modification data.

Default for trc: 0

MODTRC=trc
the table reference character (0-3) that identifies a character arrangement table specified on the CHARS parameter.

Note: The value you specify in the trc subparameter of the MODIFY parameter overrides any value you specify with the MODTRC parameter.

UCS=xxxx
an alphameric value indicating the universal character set specification (from 1 to 4 characters).

Z=nn
a numeric value (0-99) indicating a compaction table, identified by nn, is to be used when sending the SYSOUT described by this control statement to a SNA remote terminal. This specification overrides any remote-device-defined compaction table value. Setting Z to zero turns off the compaction feature. When no value is specified, the remote-device-defined compaction table is used.

Note: The Z parameter has no effect on compaction for SNA application-to-application sessions; such compaction is done only on a session basis.

General Rules for Coding

- Specifying code as "*" causes continuation of the previous OUTPUT statement, regardless of the position of the previous statement (that is, the previous statement does not have to immediately precede the continuation). The first OUTPUT statement cannot specify code as "*".
- Parameters specified on the OUTPUT statement will replace any equivalent parameters specified on the referenced DD statement.
- Code as many OUTPUT statements as you need. If more than one OUTPUT statement has the same "code" starting in column 10, the first OUTPUT statement parameters are used. If

there are duplicate parameters on the same OUTPUT statement, the last parameter overrides any preceding duplicate parameter (except on the DEST parameter).
- Use the shorter form of the parameters when coding several parameters.
- Place the OUTPUT statement after the JOB statement.
- OUTPUT statements placed after a /*ROUTE XEQ statement are ignored except at the execution node.
- OUTPUT statements placed before a /*ROUTE XEQ statement are processed on both the input and execution nodes.

When coding more than one /*OUTPUT statement:

- Any parameter coded on subsequent /*OUTPUT statements overrides the same parameter on previous /*OUTPUT statements.
- Any parameter you code on subsequent /*OUTPUT statements that you did not code on previous /*OUTPUT statements is added to the previous /*OUTPUT statement.

Rules for Coding COPIES

- If you do not specify the group value subparameter, the number of copies indicated by nnn is printed. The printed output is in page sequence for each copy.
- Parentheses are not needed when only nnn is coded.
- If you request copies of the entire job (on the JES2 JOBPARM statement) as well as additional copies of the data set (on the DD COPIES(ma parameter), and if the data set is part of the job-related output, you will receive a number of copies equal to the product of the two requests.
- The group value subparameter is for the 3800 printer only.
- Each group value can range from 1 to 255.
- You can code a maximum of eight group values on the COPIES parameter. Their sum total must not exceed 255.
- A null group value, or a zero or null position within the sublist of group values, is invalid. (For example, COPIES=(5,) and COPIES=(5,(1,0,4)) are invalid.)

Rules for Coding DEST

- If more than one destination is coded, the destinations must be in parentheses. If only one destination is coded, the parentheses are optional.
- When using the nodename.userid form of the DEST parameter, you may use continuation statements to specify up to 4 destinations. The continuation statement must contain the characters /* in columns 1 and 2, OUTPUT in columns 3 to 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      F=STD.,T=P11,C=8LPI,DEST=(FIRST.DEST)
/*OUTPUT     *         DEST=(SECOND.USER,THIRD.REMOTE)
```

Rules for Coding FCB

- If the printer on which the data set is to be printed does not have the forms control buffer feature, the operator is sent a message to mount the proper carriage control tape.
- Do not specify STD1 or STD2 unless the installation indicates that you should.

Rules for Coding FLASH

- Specify the FLASH parameter for the 3800 allocated as a system output device or for the 3800 allocated as a direct output device.
- Do not omit the overlay name. The count subparameter is example, FLASH=(ABCD,) is invalid.)
- Parentheses are not needed if count is omitted.

- The maximum number of copies that can be flashed with the with the forms overlay is determined by the value of nnn, or the sum total of group values on the COPIES parameter. If the count value of the FLASH parameter is greater than the value of the COPIES parameter, the difference is ignored.
- If you do not specify count, or you code it as 0, all copies printed are flashed with the specified overlay.

Rules for Coding INDEX and LINDEX

If the 3211 printer has the INDEX feature, it will offset the first physical print position to the right by the number of print positions specified to cause the total print line width to be reduced by the number of print positions specified. (That is, a specification of 30 will mean that the maximum line width is now 30 positions less than the original value.) These parameters are ignored on printers other than the 3211.

Rules for Coding LINECT

- Code LINECT=0 to prevent JES2 from ejecting to a new page when the number of lines exceeds the JES2 initialization parameter &LINECT.
- If LINECT is not coded on the /*OUTPUT statement, JES2 obtains the value from one of the following sources in the sequence given:
  1. The LINECT value coded on the /*JOBPARM statement
  2. The linect field of the accounting information parameter on the JOB statement
  3. The value coded in the code field of the JES2 &LINECT initialization parameter
  4. The value specified at initialization in the &LINECT statement.

Rules for Coding MODIFY

- Specify the MODIFY parameter for the 3800 that is allocated as a system output device or as a direct output device.
- Code the trc subparameter as 0, 1, 2, or 3. The value corresponds to the sequence in which the character arrangement tables are specified with the CHARS parameter, (for example, 1 refers to the second table name coded with the CHARS parameter).
- Do not omit the module name. The trc subparameter is optional, but a null position is not allowed. (For example, MODIFY=(A,) is invalid.)
- Parentheses are not needed if trc is omitted.

Example of the OUTPUT Statement

```
/*OUTPUT ABCD COPIES=6,COPYG=(1,2,3),DEST=RMT23
```

Refers to all SYSOUT data sets within the job whose DD statement specified 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 PRIORITY Statement

Control Statement

The PRIORITY statement assigns a queue selection priority to a job. This selection priority applies to all of the queues (except hardcopy) to which the job or its output might be queued. A priority value specified on a PRIORITY statement overrides any priority specified with the PRTY parameter on a JOB statement.

Note: Depending on the JES2 initialization options specified, the /*PRIORITY statement might be ignored. In a JES2 network, the /*PRIORITY statement is not ignored if it immediately follows the /*XMIT statement and precedes a valid JOB statement. When the /*PRIORITY statement does not immediately follow an /*XMIT statement, the /*PRIORITY statement is ignored at any node execept the input node.

For further information on the use of PRIORITY, see "Routing a Job Through the System (JES2)".

The PRIORITY statement consists of the characters /* in columns 1 and 2, the word PRIORITY in columns 3-10, and at least one blank followed by the priority specification. Columns 72-80 are ignored.

```
/*PRIORITY  p
```

p
a number between 0 and 15 that indicates the priority of the job.

Default: If PRIORITY is not present, or if PRIORITY is ignored, priority is derived using information from (1) the PRTY parameter on the JOB statement, (2) the accounting information on the JOBPARM statement, (3) the accounting information on the JOB statement, or (4) an installation-defined default.

Rule for Coding

- The PRIORITY statement must immediately precede the JOB statement. If it does not, or if "p" is not a number between 0 and 15, the PRIORITY statement is ignored and the input stream is flushed until a JOB statement or another PRIORITY statement is found.

Example of the PRIORITY Statement

```
/*PRIORITY  7
```

The job has a queue selection priority of 7. This value only has meaning in relation to other jobs in the system.

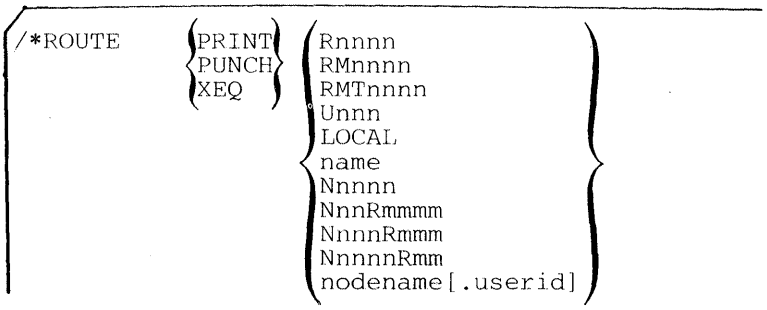
The ROUTE Statement

Control Statement

The ROUTE statement specifies the destination of the output that is not specifically routed using the DEST parameter.

The ROUTE statement consists of the characters /* in columns 1 and 2, ROUTE in columns 3-7, at least one blank followed by PRINT, PUNCH, or XEQ, then at least one blank followed by one of the device specifications. The device specification must be followed by at least one blank, which must precede column 72. Columns 72-80 are ignored.

For further information, see "Obtaining Output (JES2)."



PRINT
specifies that the job's printed output is to be routed.

PUNCH
specifies that the job's punched output is to be routed.

XEQ
specifies that the job is to be routed for execution.

Nnnnn
a node identified by nnnn which is a 1-4 digit decimal number between 1 and 1000 inclusive.

NnnRmmmm, NnnnRmmm, or NnnnnRmm
where N designates a node and R designates a remote work station that is connected to the node.
nn, nnn, or nnnn is 1-99, 1-999, or 1-1000 and mm, mmm, or mmmm is 1-99, 1-999, or 1-1000. The maximum number of digits (n and m combined) can not exceed six.
Specification of R0 is equivalent to the specification LOCAL at node N.

Rnnnn

RMnnnn

RMTnnnn
remote terminal, nnnn indicating a 1- to 4-digit numeric value defining the remote terminal to which the output is to be sent. The valid range of nnnn is 1 to the maximum number of work stations assigned to JES2. The number of work stations cannot exceed 1000.

Note: R0 is equivalent to LOCAL.

Unnn
local terminal, nnn being a 1- to 3-digit decimal value that specifies a specific local device to which the output is to be sent. The valid range for nnn is from 1 to 255, inclusive.

LOCAL
any local device at the originating node.

name
a 1- to 8-character (alphameric or national) name of a remote or local device (as defined by the system programmer) that is to receive the output.

nodename[.userid]
a node name and user or remote identifier (each from 1 to 8 characters) separated by a delimiter. The delimiter may be a period (.) as shown, a colon (:), a slash (/), or a paired parentheses (enclosing the userid). Nodename must consist of alphameric characters. Userid may contain any valid characters except the comma (,) right parenthesis ()), R, RM, RMT, or blank character. This parameter is used to route information between JES2 nodes and eligible virtual machines at VM nodes or eligible JES3 nodes.

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

Rules for Coding

- Place the ROUTE statement after the JOB statement.
- A ROUTE statement can be used to direct job execution or print or punch routing of output. However, each ROUTE statement can only contain one routing. Therefore, if execution, print, and punch are all to be routed, three statements must be used.
- If more than one ROUTE statement has the same category (PRINT, PUNCH, or XEQ), the last ROUTE statement for each category will be used.
- /*ROUTE XEQ statements are only processed on the input node and should precede any //SYSIN DD statement.
- /*ROUTE PRINT or /*ROUTE PUNCH statements following a /*ROUTE XEQ statement are only processed on the execution node. A /*ROUTE PRINT or /*ROUTE PUNCH statement in front of a /*ROUTE XEQ statement is processed on both input and execution nodes.

Examples of the ROUTE Statement

```
/*ROUTE      PRINT    RMT6
```

Routes printed output to remote terminal 6.

```
/*ROUTE      PUNCH    PUNCH2
```

Routes punched output to device "PUNCH2" as defined by the system programmer.

```
/*ROUTE XEQ DENVER
```

Routes job to node named DENVER for execution.

The SETUP Statement

Control Statement

SETUP is a control statement which is used to indicate volumes needed for executing a phase of the job.

Use of the SETUP statement in a JES2 network generally requires that the SETUP statement follow any ROUTE XEQ statement. This will prevent the setup from being requested on a node other than the node of execution. If the SETUP statement is processed prior to processing of the ROUTE or XEQ statements, setup will be requested on both the input and execution nodes.

For further information on the use of the SETUP statement, see "Routing a Job Through the System (JES2)".

The SETUP statement consists of the characters /* in columns 1 and 2, the word SETUP in columns 3-7, and at least one blank followed by the volume serial numbers. Columns 72-80 are ignored.

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

volume serial number
identifies the volume or volumes required for execution of the job.

Rules for Coding

- All SETUP statements should be placed after the JOB statement.
- As many SETUP statements as necessary can be used.

Example of the SETUP Statement

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

The two volumes requested are listed on the console when the job enters the system. The job is then placed in the hold status awaiting release by the operator when the required volumes are available. The message informs the operator that the volumes should be mounted before the job is run.

The SIGNOFF Statement

Control Statement

SIGNOFF is a control statement that indicates to the central computer that the user wishes to terminate a remote job stream processing session. At the completion of the current print and/or punch streams, JES2 disconnects the station from the system. If jobs are being read into the system from the remote station when the output is completed, JES2 disconnects the remote station when the input is completed.

Both SNA (systems network architecture) and BSC (binary synchronous communication) remote work stations can use the SIGNOFF statement. SNA remote stations, however, can also use the LOGOFF command to end a session with JES2. The LOGOFF command has some options that are not provided by the SIGNOFF statement. For a discussion of the LOGOFF command, refer to **System Programming Library: JES2 Installation, Initialization, and Tuning** and **OS/VS2 MVS System Programming Library: VTAM.**

The SIGNOFF statement consists of the characters /* in columns 1 and 2, and the word SIGNOFF in columns 3-9.

```
/*SIGNOFF
```

Example of the SIGNOFF Statement

```
/*SIGNOFF
```

Requests that a remote job stream processing session be terminated.

The SIGNON Statement

Control Statement

SIGNON is a control statement that indicates to the central computer that the user wishes to begin a remote job stream processing session. The SIGNON statement overrides the remote identification number normally assigned to the remote station. This statement is optional for all work stations except non-multi-leaving remote stations on a switched line.

Note: SNA remote work stations must use the LOGON command instead of the SIGNON statement to notify JES2 of a connection request. For a discussion of the LOGON command, refer to **System Programming Library: JES2 Installation, Initialization, and Tuning** and **OS/VS2 MVS System Programming Library: VTAM.**

The SIGNON statement consists of the characters /* in columns 1 and 2, the word SIGNON in columns 3-8, and REMOTEnnn starting in column 16. The SIGNON statement can also include two passwords: one beginning in column 25 and the other in column 73.

```
/*SIGNON         REMOTEnnnn[password1]         [password2]
```

nnnn
a value specifying the remote identification number (1 to 1000) assigned to the remote station that is requesting to sign on.

password1
a password, assigned to a dial line, that allows the remote station access to JES2 for remote job stream processing. This password is established at JES2 initialization, and can be changed or deleted by the operator with the $T command.

password2
a password that ensures that the remote station signing on is a valid RJE (remote job entry) station. This password is established at JES2 initialization.

Rules for Coding

- For multi-leaving remote stations, the SIGNON statement is put at the end of the JES2/RTP program deck.
- For non-multi-leaving remote stations, the SIGNON statement is transmitted alone as part of the initial connection process.
- Code REMOTEnnn with the same character string as the RMTnnn parameter of the ROUTE statement.

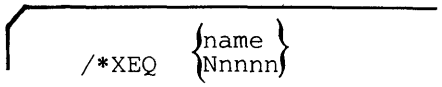
Example of the SIGNON Statement

```
/*SIGNON         REMOTE1234PSWD
```

Requests that remote station 123 begin a remote job stream processing session. PSWD, beginning in column 25, is the password assigned to a dial line.

The XEQ Statement

The XEQ statement routes a job to a node for execution. It performs the same function as /*ROUTE XEQ (see "The ROUTE Statement" earlier in this chapter). The XEQ statement consists of the characters /* in columns 1 and 2, XEQ in columns 3-5, and a 1-8 character destination node name starting in column 10.



name
Specifies a 1-8 alphameric or special-character name (specified at initialization) of the node which is to receive the job for execution.

Nnnnn
Specifies the number nnnn (1 to 1000) of the node in the network which is to receive a job for execution.

Rules for Coding

Observe the following rules for coding the XEQ statement:

- Place the XEQ statement after the job statement.
- If more than one XEQ statement is included within the job, the last XEQ statement's node name is used.
- The XEQ statement is ignored on any node other than the input node.

Example

```
/*XEQ    ATLANTA
```

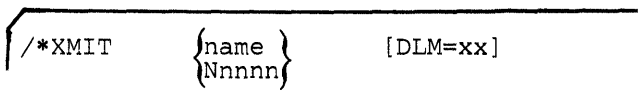
The job is to be routed to and executed on the node defined as ATLANTA. The output, by default, returns to the node of origin.

The /*XMIT Statement

Control Statement

Use the /*XMIT control statement to enter a job or data stream on a JES2 node for transmission to either another JES2 node or an eligible non-JES2 node (for example, a VM or JES3 node). JES2 will not process or check the data for JES2 validity; JES2 will only build header and trailer records for the data, and then will transmit it.

Note: All data after the /*XMIT statement is transmitted to the specified destination. The header and trailer records are built from information contained on the JOB statement preceeding the /*XMIT statement.



name
specifies a 1- to 8-alphameric or special character name (specified at JES2 initialization) of the node which is to receive the job for execution.

Nnnnn
specifies the number (nnnn) of the node in the network which is to receive a job for execution.

DLM=xx
specifies a 2 byte character string that is used to delimit the data being transmitted.

General Rules for Coding

- Place the /*XMIT statement immediately after a valid JOB statement.
- Do not continue /*XMIT statements.
- Code only one /*XMIT statement in a JOB. If the input node is specified as the node which is to receive the job for execution the data following the /*XMIT statement is considered a job for execution on the input node.

Rules for Coding DLM

- If one character or more than 2 characters are specified, the default is used.
- The delimiter can be any two characters.
- If the delimiter value specified on DLM=delimiter contains any special characters, enclose the value in apostrophes. If you include an ampersand or an apostrophe in the delimiter, you must code each ampersand or apostrophe as two consecutive ampersands or apostrophes. Any character that is neither alphameric nor national and does not appear in the special character table must be treated as if it were a special character; that is, it must be enclosed in apostrophes. Violating the special character coding rules produces unpredictable results.
- If you code the DLM parameter, the characters you assign as delimiters override any delimiter implied by the defaults. You must terminate the data with the characters you assigned in the DLM parameter.
- The characters // are not valid delimiters unless specifically indicated by DLM=//.

Defaults for DLM

- If you do not specify DLM, JES2 uses a default of /*.

- If the value specified by DLM= or /* is not encountered in a data stream being submitted on a card reader, the end-of-file condition terminates the data stream being transmitted.
- If the value specified by DLM= or /* is not encountered in a data stream being submitted through an internal reader, the close of the internal reader terminates the data stream being transmitted.

Coding JES3 Control Statements

The JES3 statements are coded with JCL statements to control the input and output processing of jobs. Rules for coding JCL, including syntax, in the section "Coding JCL Statements," apply to the JES3 statements. However, there are additional rules for coding JES3 statements. They are:

- Columns 1 through 3 always contain the characters //* (Column 4 must be a non-blank).
- JES3 statements (except command statements) can be continued by punching a comma as the last character of the first card, punching //* in columns 1 through 3 of the continuation card, and resuming the text in column 4 of the continuation card.
- JES3 control statements, except the command statement, must appear after the JOB statement, including all JOB continuation cards. JES3 statements that appear before or in the middle of the JOB statement are ignored.
- Do not place JES3 control statements in a cataloged procedure; they are ignored.
- Do not include comments on JES3 control cards (except ENDPROCESS and PAUSE statements).

The Command Statement

Control Statement

The command statement specifies JES3 operator commands, except *DUMP and *RETURN, that are entered only through the card reader or the system console. Examples in this book illustrate the format for commands entered through the card reader. Commands entered through the system console should o mit the //* from the command.

For a detailed description of the command statement and the names of the JES3 verbs and operands, see **Operator's Library: OS/VS2 MVS JES3 Commands**. There are two fields—a JES3 command starting with an asterisk in column 4 followed by one or more operands. Columns 73-80 are ignored.

```
//**command-verb     [operand ,operand. . .]
```

***command-verb**
indicates which JES3 command the operator is to perform.

CALL	X	INQUIRY	I
CANCEL	C	MESSAGE	Z
DELAY	D	MODIFY	F
DISABLE	H	RESTART	R
ENABLE	N	SEND	T
ERASE	E	START	S
FAIL		SWITCH	
FREE		VARY	V

operand
one or more operands pertaining to the command-verb.

Rules for Coding

- Begin the command-verb in column 5; code it in the same format as if it were being entered from a console.
- All command statements must precede the first JOB statement in the input stream if jobs are also being submitted. If any command statements follow the JOB statement, they are considered comments statements.
- Multiple command statements can be entered at one time.
- These statements can be placed as the first cards in an active card reader that is being restarted.
- Command statements cannot be continued from one statement to another.
- Commands that are entered on the command statement are executed immediately. They cannot be linked with any execution process of a job.

Examples of the Command Statement

```
//**VARY,280,OFFLINE
//**V,281,OFFLINE
//**VARY,282,OFF
        -or-
//**V,280-282,OFF
```

Several devices are varied offline by coding either three separate commands or one command for all devices to be varied offline. If these cards are placed in card reader 01C, for example, and it is currently not in use, the operator would then enter:

```
*X CR,IN=01C
```

```
//**MESSAGE,CN1,OUTPUT FROM JOB X REQUIRES SPECIAL CONTROLS
```

Gives a special instruction to the operation staff from a remote location. This card is placed in front of the first job.

The DATASET Statement

see page 326

Control Statement

The DATASET statement defines the beginning of an additional input stream data set that can contain JCL and/or data. The data set can be used as input to a DSP (for example, OUTSERV).

```
//*DATASET  DDNAME=ddname[,MODE=   {E}][,J={NO }]    [,CLASS=   {NO      }]
                                   {C}      {YES}                {MSGCLASS}
                                                                 {class    }
```

Defaults: MODE=E

J=NO

DDNAME

defines the ddname used to reference the spooled data set. The ddname coded in the DDNAME parameter on the //*DATASET statement can refer to only one DD statement in the job. This ddname should be qualified to the same level (stepname.procstepname.ddname), and match exactly, the ddname on the associated DD statement.

MODE

defines the card reading mode.

- E specifies that the cards are read as EBCDIC with validity checking.
- C specifies that the cards are read in card image form (column binary or data mode 2).

J

determines how the data set is terminated.

- NO indicates that a JOB statement will terminate the data set.
- YES indicates that JOB statements can be included in the data set and an ENDDATASET statement will terminate the data set.

CLASS

defines the class to be used for JES3 job processing.

- NO indicates that the system will assign a default class.
- MSGCLASS indicates the data set has the same class as the MSGCLASS of the job.
- class indicates a user-specified sysout class.

General Rules for Coding

- MODE=C is invalid for jobs read from disk or tape, and for jobs submitted from remote work stations.

Rules for Coding when MODE = C

- The J parameter is ignored when MODE=C is specified. The data set must be terminated with an ENDDATASET statement.
- You must ensure that the operator includes the C operand on the *CALL operator command for the reader that reads this job.

Examples of the DATASET Statement

```
//*PROCESS OUTSERV
//*DATASET  DDNAME=MYPRINT
            data cards
//*ENDDATASET
//*FORMAT   PR,DDNAME=MYPRINT,COPIES=5
```

Creates a data set, DDNAME=MYPRINT, and prints five copies.

```
//JOB1  JOB
//STEP1 EXEC    PROC1
//DD6   DD      UNIT=(CTC,,DEFER),VOL=SER=DUMMY,
//              DC3=(RECFM=F,LRECL=80,BLKSIZE=80)
            .
            .
            .
//*DATASET  DDNAME=STEP1.PROC1.DD6
              data cards
//*ENDDATASET
//STEP2 EXEC
```

Creates a data set, referenced by the name STEP1.PROC1.DD6, which contains the information between the //*DATASET statement and the //*ENDDATASET statement. The data set is defined by DD6 in STEP1.

The ENDDATASET Statement

Control Statement

The ENDDATASET statement terminates the creation of a JES3 input stream data set that was defined by the DATASET statement.

```
//*ENDDATASET
```

Rule for Coding

The ENDDATASET statement must appear immediately after the last card for that data set.

Example of the ENDDATASET Statement

```
//*PROCESS OUTSERV
//*DATASET  DDNAME=MYPRINT
            data cards
//*ENDDATASET
//*FORMAT   PR,DDNAME=MYPRINT,COPIES=5
```

Creates a data set, DDNAME=MYPRINT, and prints five copies.

The ENDPROCESS Statement

Control Statement

The ENDPROCESS statement indicates the end of a series of PROCESS statements for the job JCL in which it appears. No more PROCESS statements can be included in this job. The use of the ENDPROCESS statement is optional if any JCL statement follows the last PROCESS statement; otherwise, the ENDPROCESS statement must be used after the last PROCESS statement.

```
//*ENDPROCESS
```

The ENDPROCESS statement has no operands.

Example of the ENDPROCESS Statement

```
//*ENDPROCESS   END OF PROCESS STATEMENTS
```

The FORMAT Statement

Control Statement

The FORMAT statement communicates special destination and format related instructions to the system for processing the print and punch data sets.

```
//*FORMAT  {PR}
           {PU}
```

PR
indicates this card is associated with a print data set.

PU
indicates this card is associated with a punch data set.

(See page 303)
(See page 309)

General Rules for Coding

- There must be at least one blank between //*FORMAT and PR or PU.
- The text depends on the type of data set and is explained on the following pages.
- Multiple //*FORMAT statements can be used for any data set to specify special requirements for each copy of the data set.
- Classes are established at initialization that group characteristics for job output. See the system programmer to determine if you should use one of the defaults or should code the //*FORMAT statement.
- //*FORMAT statements are required for special data set processing such as multiple destinations, multiple copies of output having different attributes, forced single or double space control, and printer overflow checking. For information on special (nonstandard) jobs, see the section, "The PROCESS Statement".
- //*FORMAT statements are cumulative within a job and will apply to any //*PROCESS statement that would normally be affected by a //*FORMAT statement.

This page deleted by MVS/System Product - JES3 Release 2 5740-XYN

This page deleted by MVS/System Product — JES3 Release 2 5740-XYN.

This page deleted by MVS/System Product - JES3 Release 2 5740-XYN

This page deleted by MVS/System Product — JES3 Release 2 5740-XYN.

The PR Parameter

The PR parameter specifies print characteristics of a JES3 output data set.

For further information, see "Obtaining Output (JES3)". page 19

```
//*FORMAT        PR,DDNAME= [ddname]
                            [SYSMSG]
                            [JESJCL]
                            [JESMSG]
                [,DEST= {ANYLOCAL                                  }]
                        {device-name                               }
                        {device-address                            }
                        {group-name                                }
                        {node name [.remote]                       }
                        {LOCAL                                     }
                        {(type   [,device-name    ]      )         }
                        {        [,device-address ]                }
                        {        [,group-name     ]                }
                        {        [,LOCAL          ]                }
                [,EXTWTR=name]
                [,CHNSIZE=                    ]
                                {DS        }
                                {nnn       }
                                {(nnn,mmm)}
                [,COMPACT=compaction table name]
                [,CONTROL= {PROGRAM}     ]
                           {SINGLE }
                           {DOUBLE }
                           {TRIPLE }
                [,COPIES=   (nnn[,(group value[,group value]...)])]
                [,FORMS=   {STANDARD }   ]
                           {form-name}
                [,FCB=     {6         }  ]
                           {image-name}
                [,CARRIAGE= {6                  }  ]
                            {carriage-tape-name}
                [,TRAIN=   {STANDARD]  }
                           {train-name}
                [,THRESHLD= {Limit    }   ]
                            {99999999}
                [,CHARS=   {STANDARD                        }]
                           {(table name[,table name]...)}
                [,STACKER= {STANDARD}    ]
                           {S        }
                           {C        }
                [,FLASH=   {STANDARD                  }  ]
                           {(overlay name[,count])}
                [,MODIFY=   (module name[,trc])]
                [,OVFL={ON }  ]
                       {OFF}
                [,PRTY= nnn ]
```

Defaults:

CONTROL=PROGRAM
COPIES=1
FORMS=STANDARD
FCB=6
CARRIAGE=6
TRAIN=STANDARD
THRESHLD=99999999
OVFL=ON

DDNAME

- ddname specifies the ddname of the data set to which this statement applies. This ddname should be qualified to the same level (stepname.procstepname.ddname), and

match exactly, the ddname on the associated DD statement. (See the example for the DATASET statement.)

- SYSMSG specifies system messages.
- JESJCL specifies jclfile including statement messages.
- JESMSG specifies JES3 and system operator messages (job log).

DEST

specifies the printer used for output.

- ANYLOCAL specifies that any device (a printer as defined by the output class on the DD statement) attached to the central CPU may receive the output data set.
- device-name specifies the 1-8 alphameric or national character name of a local printer (as defined by the system programming staff) to receive the output data set.
- device-address specifies the three character physical device address of the device to receive the output data set.
- group-name specifies the name of a group of local devices, an individual remote station, or a group of remote stations to receive the output data set.
- nodename specifies a 1- to 8-character name that is the node to which JES3 networking is to direct any output.
- remote specifies a 1- to 8-character name that is a remote work station or VM userid that the receiving node directs output to.
- LOCAL specifies the default group-name for local devices (that is, those devices that are in no other group). There must be at least one local device that is not assigned to a group in order for the data set to be printed.
- type is in the formatt gggssss where ggg is the general device classification (for example, PRT and ssss is the specific device classification (for example, 3211) as defined by the system programming staff. Type must be enclosed in parentheses.
- EXTWTR specifies a 1- to 8-character name that is the name of the external writer routine, at the destination node, that is to process the SYSOUT data set.

CHNSIZE

Specifies the number of logical records to be transmitted to the work station as a systems network architecture (SNA) chain and indicates an output checkpoint is to be taken for this data set. (Valid only if transmitting to a SNA work station.)

- DS is the default and specifies that the data set is to be sent as a single SNA chain.
- nnn specifies the chain size in pages. nnn can be any decimal value from 1 through 255. The size of a page is determined by a carriage control character in the data that specifies a skip to channel 1 or by the value you assign to mmm.
- mmm specifies the number of logical records in a page when the data contains no carriage control characters. mmm can be any decimal value from 1 through 255.

Note: When CHNSIZE=DS is specified, JES3 takes normal output checkpoints.

COMPACT

Specifies the symbolic name of the compaction table to be used when the data set is transmitted to a work station. The symbolic name may contain from 1-8 characters.

CONTROL

specifies the type of forms control used.

- PROGRAM indicates that the carriage control character is the first character of each logical record in the data set.
- SINGLE indicates forced single spacing.

- DOUBLE indicates forced double spacing.
- TRIPLE indicates forced triple spacing.

COPIES

specifies the number of original copies to be printed for this data set.

- nnn can range from 0 to 255.
- group value describes the grouping of the printed copies (for the 3800 only). Each group value can range from 1 to 255.

FORMS
specifies the printer forms used.
- STANDARD indicates that standard installation forms are used.
- form-name can be from 1 to 8 characters and indicates the form name or form number of special forms to be mounted.

FCB
specifies the name of the forms control buffer used (for the 3211, 3203-5, or 3800).
- 6 specifies that the installation standard forms control buffer is used.
- image-name is a 1- to 4-character string consisting of the last characters of the FCB2xxxx member (for 3211 or 3203-5 printers or printers supported by SNA) or FCB3xxxx member (for a

CARRIAGE
specifies the carriage tape for either 3211, 3203-5, or 1403 to print onto this SYSOUT class.
- 6 specifies that the installation standard carriage tape is used.
- carriage-tape-name can be from 1 to 8 characters.

TRAIN
specifies the printer train used.
- STANDARD specifies that the installation-defined default (specified at initialization) is used.
- train-name specifies an installation-supplied name of a printer train. Check with your system programming group for installation-supplied and supported names.

THRESHLD
specifies the value to be used by output services to determine the number of data sets that are grouped together as a single unit of work for an output service writer.
- If the number specified in the THRESHLD parameter is exceeded, any excess data sets are queued as a separate unit of work.
- The value specified in this parameter overrides the value specified on the OUTSERV and SYSOUT initialization statements.

CHARS
specifies the names of character arrangement tables for printing a data set with the 3800.
- STANDARD indicates that the standard installation character arrangement table is used.
- table name is 1 to 4 alphameric or national characters identifying the name of the table.

STACKER
specifies which stacker of the 3800 Printing Subsystem the paper output is to go to.
- STANDARD indicates that the standard installation default is used.
- S indicates that the Burster-Trimmer-Stacker is used.
- C indicates that the continuous forms stacker is used.

FLASH
identifies the forms overlay to be used on the 3800.
- STANDARD indicates that the standard installation forms overlay is used on all copies printed.
- overlay name is 1 to 4 alphameric or national characters identifying the forms overlay frame to be used.
- count indicates the number of copies (from 1 to 255) to be flashed with the overlay.

See the **Forms Design Reference Guide for the IBM 3800 Printing Subsystem** for information on designing and making forms overlays.

MODIFY
specifies the name of a copy modification module used to replace blanks or data in the data set printed by the 3800. See the **IBM 3800 Printing Subsystem Programmer's Guide** for more information.
- module name is 1 to 4 alphameric or national characters identifying the module
- trc is a table reference character from 0 to 3 that corresponds to a character arrangement table specified with the CHARS parameter.

OVFL
specifies whether or not the printer program should test for forms overflow.
- ON specifies that the printer program should eject whenever the end-of-forms indicator (channel 12) is sensed.
- OFF specifies that forms overflow control is not used.

PRTY
specifies the priority at which the data set is to enter the output queue. nnn can range from 0 to 255.

Rules for Coding DDNAME

- ddname specifies the ddname of the DD statement that defines the data set that the user desires to access. This ddname should be qualified to the same level (stepname.procstepname.ddname), and match exactly, the ddname on the associated DD statement. (See the example for the DATASET statement.)
- SYSMSG specifies system messages.
- JESJCL specifies jclfile including statement messages.
- JESMSG specifies JES3 and system operator messages (job log).
- When DDNAME= is coded, the parameters specified on this FORMAT PR statement become the defaults for the job. These parameters apply to all of the job's print data sets unless overridden by a specific JES3 FORMAT control statement.

Rules for Coding DEST

- If the parameter is omitted, the first available printer in the origin group (the group of printers defined for the local or remote submitting locations) that fulfills all processing requirements is assigned. If the job originated at a remote RJP terminal, the output is returned to the originating terminal group.
- The PR DEST parameter overrides the MAIN ORG parameter.
- The nodename you specify must be a valid node defined at JES3 initialization.

Rule for Coding EXTWTR

- The external writer module specified must be defined in the remote JES3 node that will execute the job.

Rules for Coding CHNSIZE

- Use of this operand should be carefully considered because each value 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 intervention required occurs. The remote operator can eliminate duplicate data by using existing commands to reposition and restart the output writers.
- When an end-of-chain indicator is sent in the data set, JES3 takes an output checkpoint. You can provide additional checkpoints for critical data by use of the end-of-chain indicator. For example, when printing checks you can have an output checkpoint taken for each check by specifying each check as a SNA chain.

Rule for Coding CONTROL

When coding CONTROL=PROGRAM, the character specified can be either the extended USASCII code or the actual channel command code for the System/360 and System/370 channel.

Rules for Coding COPIES

- Parentheses are not needed when only nnn is coded.
- If zero is specified, printing for this data set is bypassed.
- The group value subparameter is for the 3800 only. When group values are coded, they override nnn.
- A maximum of 8 group values can be coded. Their sum total must not exceed 255.

Rules for Coding FCB

- FCB and CARRIAGE are mutually exclusive on the FORMAT statement.
- FCB should only be used when requesting a 3211 printer, a 3800 printer, a 3203-5 printer, or a printer supported by SNA RJP; otherwise it is ignored.

Rules for Coding COMPACT

- If a compaction table is not specified, is invalid, or is not known to JES3, compaction is performed using the default table. Also, a message is written to the console.
- If the remote printer does not support compaction, this parameter is ignored.

Rules for Coding CARRIAGE

- CARRIAGE and FCB are mutually exclusive on the FORMAT statement.
- For 3211 or 3203-5 printers, a module must name.

Rules for Coding TRAIN

- The TRAIN parameter should not be used for output destined for a remote RJP terminal.
- The IBM-supplied names are in the chapter "Obtaining Output (JES3)". Because these are not standard machine features, verify that the installation has the required printer train before specifying one of the parameter values.

Rule for Coding CHARS

- From 1 to 4 table names can be specified. Parentheses are not needed when one table name is coded.

Rules for Coding FLASH

- Parentheses are not needed if count is omitted.
- If count is not specified or coded as 0, all copies printed are flashed with the specified overlay.

Rules for Coding MODIFY

- Parentheses are not needed if trc is omitted.
- If trc is not specified, the first character arrangement table coded with the CHARS parameter is used for printing the data.
- The copy modification module is defined and stored using the IEBIMAGE utility program.

Rule for Coding OVFL

For remote job processing, the overflow test is a responsibility of the terminal package for the remote RJP terminal. OVFL is ignored for remote job processing. For additional information on the use of OVFL, see the topic "Requesting Forms Overflow and Printer Spacing" in this publication.

Examples of the Print Parameter

```
//*FORMAT    PR,DDNAME=REPORT,COPIES=2
```

Specifies that two original copies of the data set defined by REPORT are requested. Any printer with standard forms, train, and carriage tape mounted can be used.

```
//*FORMAT    PR,DDNAME=,DEST=ANYLOCAL
```

Specifies that all data sets without FORMAT statements are printed on any local printer.

The PU Parameter

The PU parameter specifies punch characteristics of a JES3 output data set.

For further information, see "Obtaining Output (JES3)".

```
//*FORMAT    PU,DDNAME=ddname
             [,DEST=(ANYLOCAL                                    )      ]
                    (device-name                                 )
                    (device-address                              )
                    (group-name                                  )
                    (nodename[.remote]                           )
                    (LOCAL                                       )
                    (type  [,device-name    ]      )              )
                           [,device-address]                      )
                           [,group-name    ]                      )
                           [,LOCAL         ]                      )
             [,EXTWTR=name]
             [,CHNSIZE=      {DS        }   ]
                             {nnn       }
                             {(nnn,mmm)}
             [,COMPACT=compaction table name]
             [,COPIES=   nnn  ]
             [,FORMS=    {STANDARD }   ]
                         {form-name}
             [,INT=  {YES}]
                     {NO }
```

Defaults: COPIES=1
FORMS=STANDARD
INT=NO

DDNAME

specifies the ddname of the data set to which this statement applies. This ddname should be qualified to the same level (stepname.procstepname.ddname), and match exactly, the ddname on the associated DD statement. (See the example for the DATASET statement and the "Rules for Coding DDNAME" under the PR parameter description.)

DEST

specifies the punch unit used for output.

- ANYLOCAL specifies that any device (a punch as defined by the output class on the DD statement) attached to the central CPU may receive the output data set.
- device-name specifies the 1-8 alphameric or national character name of a local punch (as defined by the system programming staff) to receive the output data set.
- device-address specifies the three character physical device address of the device to receive the output data set.
- group-name specifies the name of a group of local devices, an individual remote station, or a group of remote stations to receive the output data set.
- nodename specifies a 1- to 8-character name that is the node to which JES3/networking is to direct any output
- remote specifies a 1- to 8-character name that is a remote work station or VM userid that the receiving node directs output to.
- LOCAL specifies the default group-name for local devices (that is, those local devices that are in no other group). There must be at least one local device that is not assigned to a group in order for the data set to be printed.
- type is in the format gggssss where ggg is the general device classification (for example, PUN and ssss is the specific device classification (for example, 3525) as defined by the system programming staff. Type must be enclosed in parentheses.

EXTWTR

specifies a 1- to 8-character name that is the name of the external writer routine that is to process the SYSOUT data set at the destination node.

CHNSIZE

Specifies the number of logical records to be transmitted to the work station as a systems network architecture (SNA) chain and indicates an output checkpoint is to be taken for this data set. (Valid only if transmitting to a SNA work station.)

- DS is the default and specifies that the data set is to be sent as a single SNA chain.

- nnn specifies the chain size in pages. nnn can be any decimal value from 1 through 255. The size of a page is determined by a carriage control character in the data that specifies a skip to channel 1 or by the value you assign to mmm.
- mmm specifies the number of logical records in a page when the data contains no carriage control characters. mmm can be any decimal value from 1 through 255.

Note: When CHNSIZE=DS is specified, JES3 takes normal output checkpoints.

COMPACT
Specifies the symbolic name of the compaction table to be used when the data set is transmitted to a work station. The symbolic name may contain from 1-8 characters.

COPIES
specifies the number of copies of the data set that are to be punched. nnn can range from 0 to 255.

FORMS
specifies the card forms used.
- STANDARD specifies that installation forms defaults are used.
- form-name can be from 1 to 8 characters and indicates the card forms to be used.

INT
specifies whether or not the output is interpreted.
- YES causes an attempt to obtain a device type PUN3525I (a card punch with the interpret feature). If DEST does not include a 3525I, INT=NO is substituted.
- NO causes no interpreting of these cards.

Rules for Coding DDNAME

- The ddname should be qualified to the level required. (See also "Rules for Coding DDNAME" under the PR parameter description.)
- When DDNAME=, is coded, the parameters specified on this statement apply to all punch data sets that have no FORMAT statements.

Rules for Coding DEST

- If the parameter is omitted, the first available punch unit in the job origin group (the group of punches defined for the local or remote submitting locations) that fulfills all processing requirements is assigned. If the job originated from a remote RJP terminal, the output is returned to the originating terminal group.
- The PU DEST parameter overrides the MAIN ORG parameter.
- The nodename you specify must be a valid node defined at JES3 initialization.

Rules for Coding CHNSIZE

- Use of this operand should be carefully considered because each value 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 intervention required occurs. The remote operator can eliminate duplicate data by using existing command 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 use of the end-of-chain indicator. For example, when printing checks you can have an output checkpoint taken for each check by specifying each check as a SNA chain.

Rules for Coding COMPACT

- If a compaction table is not specified, is invalid, or is not known to JES3, compaction is performed using the default table and a message is written to the console.
- If the remote printer does not support compaction, this parameter is ignored.

Rule for Coding COPIES

If zero is specified, punching for this data set is bypassed.

Rule for Coding EXTWTR

- The external writer module specified must be defined in the remote JES3 node that will execute the job.

Example of the Punch Parameter

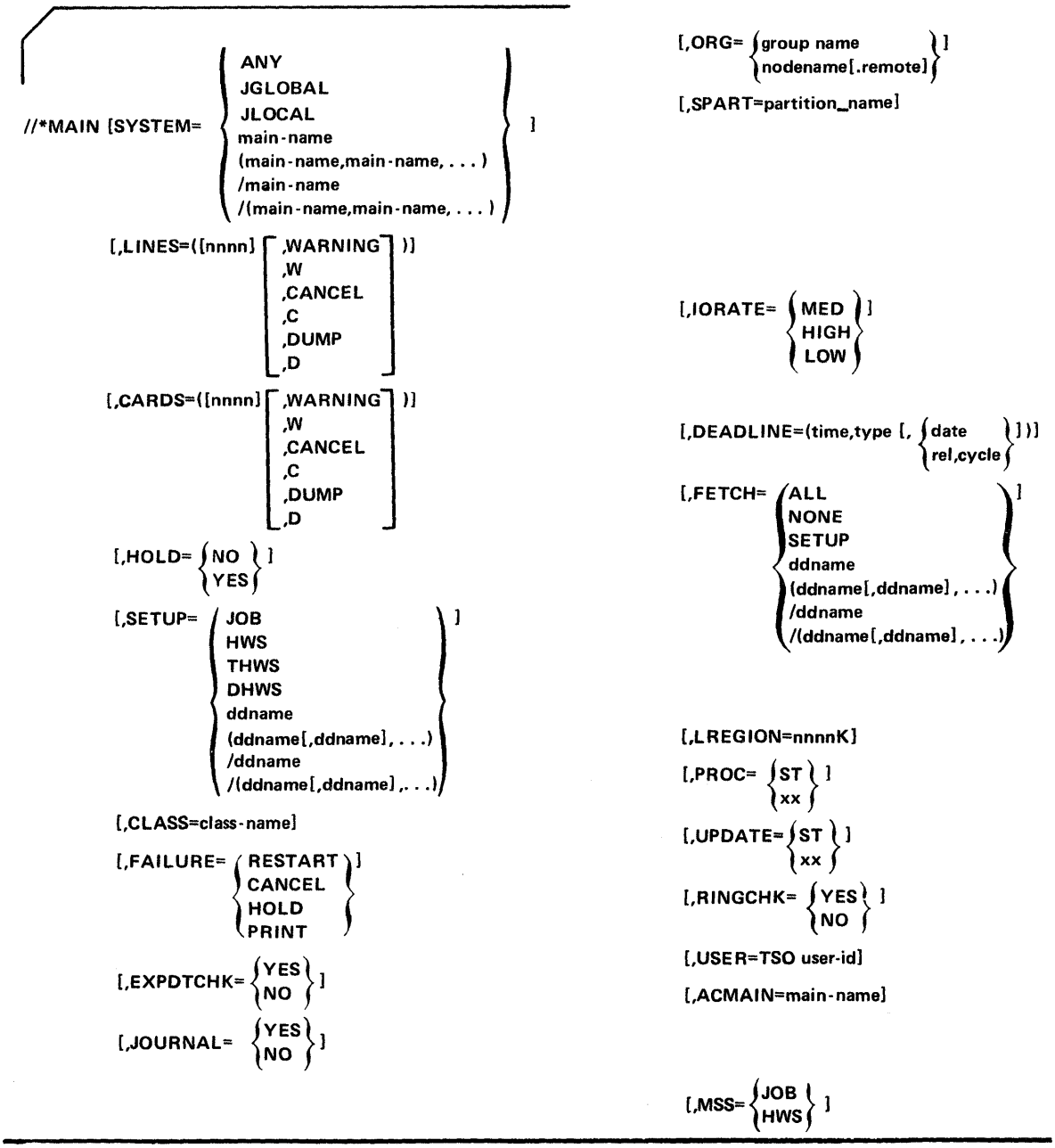
```
//*FORMAT    PU,DDNAME=PUNCHOUT,DEST=PU1,FORMS=RED-STRP
```

One copy of the data set defined by PUNCHOUT is punched on unit PU1. Before processing, the operator is requested to insert "RED-STRP" cards into the designated punch.

The MAIN Statement

Control Statement

The MAIN statement defines the processor requirements for the current job. Many of the parameters are used to override parameters of the STANDARDS initialization control card.



Defaults:

- SYSTEM=ANY
- HOLD=NO

- RINGCHK=YES
- PROC=ST
- EXPDTCHK=YES
- If you do not specify the SYSTEM parameter, the job is eligible to run on those processors for which the related job class is eligible.
- If SETUP is not specified, the device requirements for mountable tape and disk volumes is based on an initialization parameter.
- If LINES, CARDS, FETCH, or IORATE are not specified, the installation default for this job class will be used.
- If CLASS or LREGION are not specified, JES3 will determine the value of LREGION based on initialization parameters.
- If JOURNAL is not specified, JES3 will determine the value of JOURNAL based on an initialization parameter.
- If SPART is not specified, JES3 allocates spool data sets to a specific spool partition according to the following priorities:
  - The spool partition associated with the job's output class and defined by a JES3 SYSOUT initialization statement
  - The spool partition associated with the job's class and defined by a JES3 CLASS initialization statement
  - The spool partition associated with the executing processor and defined by a JES3 MAINPROC statement
  - The default spool partition

Note: For output data sets the spool partition defined by a JES3 SYSOUT initialization statement overrides all other spool partitions specified.

SYSTEM

specifies the processor by name or class of system used for this job. If a specific processor is named, the processor name must be specified on the CLASS initialization statement for the job class.

- ANY defaults to any system (global or local) that satisfies the job's requirements.
- JGLOBAL specifies that the job is to run on the global processor only.
- JLOCAL specifies that the job is to run on a local processor only.
- main-name specifies the specific processors considered for this job.
- /main-name specifies the specific processors excluded from consideration for this job.

LINES

specifies the estimated number of lines of data printed for this job. The second group of subparameters specifies the action taken when the line estimates are exceeded.

- nnnn is a 4 digit value indicating the number of lines in thousands.
- WARNING issues an operator warning and continues processing.
- CANCEL cancels the job.
- DUMP cancels the job with a storage dump.

CARDS

specifies the estimated number of cards punched for this job. The second group of subparameters specifies the action taken when the card estimates are exceeded.

- nnnn is a 4 digit value indicating the number of cards in hundreds.
- WARNING issues an operator warning and continues processing.
- CANCEL cancels the job.
- DUMP cancels the job with a storage dump.

HOLD

- YES specifies that the job will enter into the system in operator hold status and will be withheld from processing until the operator requests its release. However, if an error occurs during input service processing, the job will not be held for operator intervention

but will be scheduled for reader-interpreter/converter-interpreter processing. This parameter has the same function as TYPRUN=HOLD on the JOB statement.

- NO specifies that the job enters the system normally and processing does not require operator intervention.

SETUP

modifies the standard setup algorithm used in assigning devices to a job prior to its execution.

- JOB indicates that device requirements for mountable units are calculated by the system for the entire job.
- HWS requests allocation of the minimum number of devices required to run the job (high watermark setup).
- THWS requests high watermark setup for tapes and job setup for disks. Job setup requests units for every unique volume in the job.
- DHWS requests high watermark setup for disks and job setup for tape.
- ddname specifies the DD statements (fully qualified) that are set up before a job enters execution (explicit setup).
- /ddname removes the specified DD statements from consideration for setup (explicit setup).

CLASS
specifies the job class for this job. class-name can be 1 to 8 characters.

FAILURE
specifies the job recovery option used in case of system failure.
- RESTART restarts the job when the failing processor is restarted.
- Do not use FAILURE=RESTART for jobs using the DEQ at DEMOUNT facility for tape volumes.
- CANCEL cancels the job for printing.
- HOLD holds the job for restart.
- PRINT prints the job and then puts the job in hold for restart.

EXPDTCHK
- YES indicates that JES3 Main Device Scheduling (MDS) volume mount verification is to perform expiration date checking for scratch SL output tape volumes.
- NO indicates that expiration date checking is to be bypassed for this job.

JOURNAL
- YES indicates that the job is to have a job journal.
- NO indicates that the job is to have no job journal.

IORATE
specifies the I/O to CPU ratio for a job. This is used in an attempt to balance the mixture of jobs selected for execution on the processor. This parameter overrides the installation defined job selection criteria for this job's class.

ORG
specifies an override of the origin group name or nodename of the device used to enter the job on the JES3 system. This parameter causes any output to be directed to the specified origin group or networking node. Normally, output from a job is directed to the same group of devices or node from which it originated.

- group name parameter causes JES3 to direct any output to the specified origin group
- nodename specifies a 1- to 8-character name that JES3 networking uses as the origin of the job
- remote specifies a 1- to 8-character name that a remote workstation or VM userid that JES3/networking uses as the origin of the job

DEADLINE
specifies when the job is due to be scheduled.

- time indicates the deadline time.
  - nM specifies that the job is to be scheduled within n minutes. n can range from 0 to 1440.
  - nH specifies that the job is to be scheduled within n hours. n can range from 0 to 24.
  - hhhh specifies the time of day that the job is to be scheduled, in 24-hour clock time (0800 is eight a.m.). hhhh can range from 0 to 2400.
- type is a single character identifier (A-Z, 0-9), defined by the installation, that specifies one of the installation deadline types. If the type is not defined, the job is flushed.
- date, in the format MMDDYY for month, day, and year, specifies the date when the time parameter takes affect. If the date and rel, cycle is omitted, the current date is assumed provided that the current time is earlier than the deadline time. If the current time is later, the next day's date is assumed.
- rel is an integer from 1 to 366 that specifies which day within the cycle the deadline falls.
  - values coded with WEEKLY default to 7 if greater than 7 is specified. Sunday is day 1; Saturday is day 7.
  - values coded with MONTHLY default to 31 if greater than 31 is coded. 29, 30, and 31 are treated as the last day of the month.
  - values coded with YEARLY default to 365 for all non-leap years if greater than 365 is specified. Leap year defaults to 366.
- cycle is specified as WEEKLY, MONTHLY, or YEARLY to indicate periodic runs. For example, (1, WEEKLY) specifies that the job reaches its deadline on Sunday of every week.

FETCH
specifies an override of the installation defined FETCH parameter and determines which fetch messages will be issued to the operator for disk and tape volumes for this job.

- ALL specifies that all volumes in DD statements using JES3 setup devices, except permanently resident volumes, should be fetched.
- NONE specifies no fetch messages.
- SETUP specifies that volumes in DD statements specified in the MAIN SETUP parameter should be fetched. If no SETUP parameter is specified, the FETCH default is ALL.
- ddname specifies that only the volumes in the DD statements specified are fetched.
- /ddname specifies that all volumes in the DD statements specified are not fetched.

LREGION
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 an MVS main processor.

PROC

defines the private library searched for the catalog procedure to run the job.

- ST specifies that the standard default procedure library is searched.
- xx specifies the last 2 characters of the ddname of the additional procedure library to be searched. xx is defined by the installation. If this parameter is used, only the private library is searched; the standard procedure library is not searched.

UPDATE

defines the procedure library updated by this job. This parameter causes all jobs using this library to be held until the update is complete. If concatenated data sets are defined, the library update facility cannot be used for any of these data sets.

- ST specifies that the standard default procedure library is updated.
- xx specifies the last 2 characters of the ddname of the additional procedure library to be updated. xx is defined by the installation. If this parameter is used, only the private library is updated; the standard procedure library is not updated.

RINGCHK

- YES indicates that a validation check is to be made to determine the correct status of the tape reel ring for tape devices set up by JES3.
- NO indicates that ring checking is to be by-passed for this job.

USER

identifies the job with the specified TSO user even though the job was not submitted from TSO by that user. This parameter allows:

- SYSOUT data sets to be sent to an MVS main processor (see ACMAIN) for use by the TSO user.
- the TSO user on an MVS processor (global or local) to access SYSOUT data sets with the TSO OUTPUT command.
- the TSO user on any processor to inquire about the status of the job or to cancel the job.

ACMAIN

identifies the job with the specified processor even though the job was not submitted from or executed on that processor. This parameter allows:

- SYSOUT data sets to be sent to an MVS main processor (see USER).
- the job termination message IAT6108 to be sent to the processor if the NOTIFY parameter was coded on the JOB card.

MSS

specifies whether or not MSS processing is requested.

- JOB indicates that each 3330V unit requested by a job will be assigned a separate unit.
- HWS indicates that 3330V units will be reused in subsequent job steps in order to minimize 3330V unit usage.

The MSS parameter is to MSS devices what the SETUP parameter is to non-MSS devices. The 'JOB' and 'HWS' options are independent across device types in that 'HWS' may be specified for non-MSS devices with 'JOB' specified for MSS devices or vice versa.

SPART

specifies the spool partition that JES3 will use to allocate spool space to this job. Where:

- partition name is the name of the spool partition and must match the NAME parameter of a JES3 SPART initialization statement
- This parameter will override all other spool partition definitions for this job except the SPART parameter on the SYSOUT initialization statement for output data sets.

Rules for Coding SYSTEM

- If the control program is inconsistent with the SYSTEM parameter, the job will be flushed. The SYSTEM parameter must also be consistent with PROCESS statements.
- You do not always need to code the SYSTEM parameter because the installation often establishes defaults according to job class. Check this with the system programmer.

Rule for Coding LINES

LINES=0 only applies to jobs in the execution phase. After the first line is sent out for writing, the job will issue a warning, a cancel, or a dump, depending on what is requested. When printing the output, however, LINES=0 is ignored.

Rule for Coding CARDS

CARDS=0 only applies to jobs in the execution phase. After the first line is sent for punching, the job will issue a warning, a cancel, or a dump, depending on what is requested. When punching the output, however, CARDS=0 is ignored. The SYSABEND and SYSUDUMP SYSOUT data sets are not considered in the total line count of a job when determining whether a job has exceeded the maximum lines specified.

Rules for Coding SETUP

- When specifying ddname, enough devices must be specified to allow JES3 allocation for the maximum number of devices required at any one time. If specified devices are insufficient, the job is canceled.
- When specifying a subparameter list of ddnames that cannot be contained on a single statement, the SETUP parameter must be repeated on the continuation statement.

Rules for Coding CLASS

- If a single character class-name is used, it may be specified on the JOB statement.
- A valid CLASS parameter on the MAIN statement overrides a valid CLASS parameter on the JOB statement.

Rule for Coding EXPDTCHK

If EXPDTCHK=YES, the SL tape scratch requests that are pre-mounted by MDS must have expired labels.

Rule for Coding JOURNAL

The JOURNAL parameter overrides what was specified on the CLASS card of the JES3 initialization deck.

Rules for Coding ORG

- The FORMAT statement will override the ORG parameter if specified for the particular data set.
- The MAIN ORG statement should precede all FORMAT statements that do not contain the DEST parameter. If it does not, the default for these data sets is where the job entered the system.
- When specifying ORG on a //*MAIN statement that is part of a remote job, place the //*MAIN statement immediately after the second JOB statement.

Rule for Coding DEADLINE

If the current date is specified and the job is submitted after the deadline time, all of the priority changes are applied to make the job the same priority level it would have been if it had been submitted prior to the deadline and not completed.

Rule for Coding FETCH

When specifying a subparameter list of ddnames that cannot be contained on a single statement, the FETCH parameter must be repeated on the continuation statement.

Rule for Coding LREGION

Consult the system programming staff for guidance in using the LREGION parameter. If the values selected for LREGION are too small, the job may run slower.

Rule for Coding PROC

If a procedure is specified for which there is no corresponding PROCLIB entry, the job is flushed.

Rules for Coding UPDATE

- If a procedure is specified for which there is no corresponding PROCLIB entry, the job is flushed.
- If this parameter is specified, SYSTEM=JGLOBAL must also be specified or the job is flushed.

Rules for Coding SPART

- The //*MAIN statement containing the SPART parameter must be placed before the first EXEC JCL statement and any JES3 //*DATASET or //*PROCESS statements in the job's input stream. If the //*MAIN statement containing the SPART parameter is out of position, JES3 will use the default spool partition and issue a warning message. (See note below.)
- partition name is a 1- or 8-character name that must match the partition name specified in the NAME parameter of a JES3 SPART initialization statement. If there is an error, JES3 uses the default spool partition and issues a warning message. (See note below.)

Note: When JES3 assigns spool data sets to a spool partition, the spool data sets are assigned using specific priorities. For a discussion of these priorities see "Spool Partition" in this publication.

Rules for Coding ACMAIN

- When used for SYSOUT data sets, this parameter applies to all eligible output of the job. However, it can be overridden for a specific data set with the MAIN parameter on a FORMAT AC statement for that data set.
- This parameter is also used in association with the NOTIFY parameter on the JOB statement to indicate the TSO main processor to be notified at the completion of a job.

Example of the MAIN Statement

```
//*MAIN SYSTEM=SY1,LINES=(5,C),SETUP=HWS,
//*FAILURE=RESTART,DEADLINE=(0800,A,3,WEEKLY)
```

The job executes on system SY1. It is estimated to produce 5000 lines of printed output and if the output exceeds 5000 lines, the job is canceled. HWS specifies that a minimum number of devices required for this job is allocated. In the event of a system failure, the job is restarted on the main processor SY1. JES3 attempts to complete this job by 8 a.m. every Tuesday (Tuesday is day number 3) by adjusting the job's scheduling priority using the installation-defined A type deadline scheduling parameters.

The NET Statement

Control Statement

The NET statement defines the dependencies between jobs in a dependent job-net.

For more information, see "Dependent Job Control" earlier in this book.

```
//*NET {NETID}
       {ID   }  =name
      {NHOLD}
    [, {HC   }  =n]
      {RELEASE}
    [, {RL   }=(jobname1,jobname2,...jobnamen)]
      {NORMAL}     {D}
    [, {NC   }  = {F}  ]
                   {R}
      {ABNORMAL}   {D}
    [, {AB    }= {F}  ]
                   {R}
      {OPHOLD}     {NO}
    [, {OH   }  = {YES}]
      {RELSCHCT}
    [, {RS    }    =n]
      {NETREL}
    [, {NR   }     =(netid,jobname)]
              {ANY}
    [,DEVPOOL=({NET},device-name,number[,device-name,number,...]
               [,SDGxx,...])]
              {YES}
    [,DEVRELSE={NO }]
      NRCMP        {    NOHO  }
    [,  PC       = {    HOLD  } ]
                   {    FLSH  }
      ABCMP        {    NOKP  }
    [,  AC       = {    KEEP  } ]
```

Defaults: NORMAL=D
ABNORMAL=R
OPHOLD=NO
DEVPOOL=ANY
DEVRELSE=NO
NRCMP=HOLD
ABCMP=NOXP

NETID
specifies the name of the job-net containing this job. name can be from 1 to 8 characters; the first character must be alphabetic.

NHOLD
specifies the number of immediate predecessor job completions required before this job can be released for scheduling. n can range from 0 to 32,767.

RELEASE
specifies the jobnames of jobs in the specified job-net that are successors to this job.

NORMAL and ABNORMAL
specifies the action taken for this job when any predecessor normally or abnormally completes execution.

- D means decrement the NHOLD count (the number of predecessors) of this job. If the NHOLD count goes to zero, this job becomes eligible for scheduling.

- F means flush the job and its successors from the system. The job is canceled, any output printed, and all successors presently in the system are canceled regardless of their normal or abnormal specifications. All jobs entered after the flush have been processed, that would have been automatically flushed, are entered into the system normally. To flush these jobs, the operator must cancel the jobs or the net.
- R means retain this job in the system and do not decrement the NHOLD count. This suspends the job and its successors from scheduling until either the predecessor job is resubmitted or the operator decreases the NHOLD count.

OPHOLD
- NO specifies that the job is to be processed normally without operator intervention.
- YES specifies that the job is placed in DJC operator hold. This prevents scheduling of this job until it is explicitly released from DJC operator hold by the operator.

RELSCHCT
controls early set up of a dependent job's resources. n can range from 1 to 32,767. Set up begins when the NHOLD count becomes less than or equal to n.

NETREL
specifies that this job is a predecessor to a job in another job-net.
- netid identifies the NETID name of the successor job.
- jobname identifies the name of the successor job.

DEVPOOL
specifies devices to be dedicated to this dependent job control net.
- ANY indicates that jobs in the net can use any dedicated or non-dedicated device.
- NET indicates that jobs can use only devices dedicated to the net.
- device-name and number indicates the name and number of dedicated devices. device-name can be an installation defined name (names defined to JES3 by the installation), or any name specified in the UNIT parameter except unit address. number can range from 1 to 32,767.
- SDGxx specifies the MSS staging drive group(s) to be pooled (fenced) for the DJC network. If SDGxx is used, no quantity specification should follow because one staging drive group is assumed.

DEVRELSE
- YES specifies that all devices dedicated to the dependent job control net should be released at the end of this job.
- NO specifies that all dedicated devices are released when the last job in the net ends.

NRCMP
specifies that a DJC job that has completed normally is being resubmitted and does not enter the net and all references to the net are erased from the JCT by input service.
- HOLD means that input service is to put the job in operator hold.
- NOHO means that input service is to allow the job to be scheduled as the system resources become available.
- FLSH means that input service is to flush the job.

ABCMP
specifies what action is to be taken by the DJCUPDAT DSP if the job abnormally terminates.
- KEEP means that the net is to be kept in the system until either the job is resubmitted and completes normally or the operator forces the net from the system. Specifying this keyword ensures that the net is not purged until proper action is taken by the operator.
- NOKP means that the net is to be purged if the job that abnormally terminated has not been resubmitted by the time the other jobs in the net have completed, provided there are no missing successors and subnets.

Note: Any job that abnormally terminates can be resubmitted to the net and the net will be retained until the job completes.

General Rules for Coding

- Only one NET statement can be defined for each job of a job-net.
- The parameters on the NET statement can be coded in any order.
- The RELEASE parameter is the only parameter on the NET statement that can be split and continued.
- If a PROCESS statement is used in a job, the NET statement must come before it.

Rules for Coding NETID

- All jobs put into the system with the same NETID name form a dependent job control (DJC) net.
- NETID names must be unique within the JES3 system. A job that has the same NETID name as an existing job-net is added as a member of that job-net.

Rules for Coding NHOLD

- If n is zero or is not specified, then this job has no predecessors and is immediately eligible for scheduling.
- If an incorrect NHOLD count is specified, two situations can occur:
  1. If n is greater than the actual number of predecessor jobs, then this job is not released from DJC hold when all of its predecessors complete execution.
  2. If n is less than the actual number of predecessor jobs, this job is prematurely released from DJC hold.

Rules for Coding RELEASE

- From 1 to 50 successor jobnames can be specified.
- RELEASE values can be split on a NET continuation statement.

Rules for Coding RELSCHCT

- If n is zero or is not specified, there is no early set up of dependent jobs.
- This parameter must not be specified for a job that may have catalog dependencies in dependent job control.
- Do not specify RELSCHCT for nonstandard DJC jobs. Nonstandard DJC jobs are explained in the section, "The PROCESS Statement".

Rule for Coding NETREL

The NETREL parameter can be specified only once for each job of a given job-net.

Rules for Coding DEVPOOL

- This parameter is only recognized when found in the first job of a job-net entered into the system.
- The first subparameter indicates what devices are eligible for volume mounting by jobs in the net. If ANY is specified, allocation is attempted from the dedicated pool before any non-dedicated devices are used.
- The device-name and number can be repeated for additional devices eligible for volume mounting up to a maximum that will fit on one card.

Rule for Coding DEVRELSE

This parameter can be specified on one or more jobs in the net. The first completing job that has specified DEVRELSE=YES causes the devices dedicated to the net to be released.

Examples of the NET Statement

```
//*NET  ID=NET01,NHOLD=0,AB=F,DEVPOOL=(,3330,2)
```

Defines a job-net named NET01. There are no predecessor jobs. In the event the job fails, all successor jobs in NET01 are flushed. The DEVPOOL parameter (which must be coded with the first job in the net) requests a device pool of two 3330s to be established for the job-net.

```
//*NET  ID=N1,RL=B,NR=(N2,B2),DEVPOOL=(NET,3330,1)
```

Defines a job-net named N1. This job is a predecessor of job B, which is in N1, and job B2, which is in the job-net named N2. The RL parameter releases job B; the NR parameter releases job B2. The DEVPOOL parameter specifies that one 3330 is to be dedicated to this job-net; jobs in the net must wait for the availability of this device if it is in use.

The NETACCT Statement

Control Statement

The NETACCT statement allows you to specify accounting information to be transmitted with a job to another node in the network.

```
//*NETACCT   [PNAME=programmer's name]
             [,ACCT=nnnnnnnn]
             [,USERID=nnnnnnnn]
             [,DEPT=nnnnnnnn]
             [,BLDG=nnnnnnnn]
             [,ROOM=nnnnnnnn]
```

PNAME
1 to 20 characters that specify the programmer's name

ACCT
1 to 8 characters that specify the network account number

USERID
1 to 8 characters that specify the user in the network

DEPT
1 to 8 characters that specify the department number in the network

BLDG
1 to 8 characters that specify the building number in the network

ROOM
1 to 8 characters that specify the room number in the network

Default: The defaults for the JES3/networking header are installation specified at JES3 initialization.

General Rules for Coding

- When coding a //*NETACCT statement you must code at least one parameter
- The //*NETACCT statement must immediately follow the first JOB card and precede any //*ROUTE XEQ statements
- //*NETACCT statements cannot be continued. (If continuation is necessary, use multiple //*NETACCT statements.)

Rule for Coding PNAME

- Enclose the character string in single quotes if it contains embedded blanks or special characters.

*Example of the //*NETACCT Statement*

```
//*NETACCT    PNAME=COLLINS,ACCT=D588706,USERID=NXT
```

The OPERATOR Statement

Control Statement

The OPERATOR statement transmits any desired message to the operator. Columns 1 through 80 are sent to the LOG console when the job enters the JES3 queue.

```
//*OPERATOR text
```

Example of the OPERATOR Statement

```
//*OPERATOR CALL EXT.   641 WHEN THIS JOB STARTS
```

The PAUSE Statement

Control Statement

An input reader can be halted temporarily by punching the pseudo command, PAUSE, starting in column 5. The PAUSE statement can be entered through any reader. The reader then issues a message and waits for operator reply. The use of the PAUSE statement is intended primarily for system checkout and test. It is recognized only if submitted before the JOB statement in the input stream. It is recommended for remote users only. When the //**PAUSE statement is entered from a remote work station, another work station (with console level 15) must send the start message or the JES3 operator must issue the *START command to start the job.

```
//**PAUSE
```

Rules for Coding

- At least two blanks must follow the word PAUSE before comments are added.
- The PAUSE statement can be entered through any reader.

Example of the PAUSE Statement

```
//**PAUSE
```

The PROCESS Statement

Control Statement

The use of the PROCESS statement causes JES3 to bypass its standard job flow. Any job that includes PROCESS statements receives only the JES3 processing specified on the PROCESS statements, except that which JES3 must perform. By using one or more PROCESS statements, you can bypass JES3 job execution or bypass JES3 output processing, or both. Bypassing execution allows checking the JCL of a job, and bypassing output processing permits the operator to check (by inquiry command) whether the job reaches execution.

The PROCESS statement can also be used to call any Dynamic Support Program (DSP) that is defined in the DSP dictionary as capable of being processed. For non-standard DJC jobs, a //*PROCESS DJCPROC statement is required only when a //*PROCESS MAIN statement is not included in the job stream.

```
//*PROCESS  dsp
```

dsp
specifies the name of the DSP that is to be processed. Figure 28 lists the valid DSP names and whether parameters can follow.

Note: Input service error messages that indicate a job will be scheduled for interpreter processing prior to being purged will be scheduled for an interpreter DSP only if it is specified on the first //*PROCESS statement.

Rules for Coding

- Each PROCESS statement can have only one operand, which is the name of a DSP. Figure 28 lists the permissible DSPs.
- If the DSP requires specification of parameters, these parameters are listed on the next statement, starting in column 1, separated by commas.
- Several PROCESS statements may follow each other or be separated by parameters. JES3 accomplishes the processing for the PROCESS statements in the order they appear in the input stream.

Examples of the PROCESS Statement

```
//EXAM1          JOB
//*PROCESS       CI
//*PROCESS       MAIN
//*PROCESS       OUTSERV
//S1             EXEC
    remaining JCL statements
```

This is an example of submitting a simple job via //*PROCESS statements. It executes the same as a standard job without JES3 control statements. Four scheduler elements are created for the job. The CI DSP creates the control blocks for MAIN, the next scheduler element. The OUTSERV DSP is scheduled after MAIN completion. PURGE is the last function in any job, and has a scheduler element created automatically.

```
//EXAM2          JOB
//*PROCESS       CI
//*PROCESS       MAIN
//*PROCESS       OUTSERV
//*PROCESS       PLOT
//*PROCESS       TT
IN=(TA9),MDI=T,ID=DEP836,FILES=2
//*ENDPROCESS
//S1             EXEC
//DD1            DD      UNIT=24009,DISP=(NEW,KEEP)
    remaining JCL statements
```

This is an example using user-written DSPs and JES3 utilities. PLOT is a user-written DSP and is to be executed after output service has completed. The TT DSP is a tape-to-tape utility and is followed by the parameters needed for it to be used.

```
//EXAM3          JOB
//*PROCESS       OUTSERV
//*FORMAT        PR,DDNAME=DS1,COPIES=5
//*DATASET       DDNAME=DS1
    any data
//*ENDDATASET
```

This example uses JES3 output service and the DATASET statement. Five copies of data set DS1 are printed on any local (not remote) printer.

The ROUTE XEQ Statement

Control Statement

The ROUTE XEQ statement specifies the node in a network where the job is to execute.

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

nodename
specifies the name of the node where the job is to execute. The nodename refers to the JES3 global processor and all of its local processors.

.vmuserid
specifies the name of a VM/370 node where the job is to execute

Rules for Coding

- //*ROUTE must begin in column one
- Place the //*ROUTE XEQ statement after a valid JOB statement for the submitting location.
- Place the //*ROUTE XEQ statement after any //*NETACCT statements.
- Place the //*ROUTE XEQ statement immediately before a valid JOB statement for the executing location.
  Note: TSO users must code NJB in place of JOB on the second JOB statement.
- Do not code for nodename a system name that refers to an individual processor (local or global) within a JES3 node.
- Do not code a nodename that is the same as the local node. If the nodename is the same as the local node, the job executes locally.
- Code .vmuserid only when an operating system that is running as a virtual system at a VM/370 node is to execute the job.
- Do not code imbedded blanks in the nodename or vmuserid parameters

*Example of the //*ROUTE XEQ Statement*

```
//JOBA    JOB      (for submitting location)
//*ROUTE  XEQ   nodename
//JOBB    JOB      (for executing location)
  .
  .
  .
/*
```

JOBA is a valid JOB statement that allows the job entry into the local JES3 system. The //*ROUTE XEQ statement causes JES3/networking to divert the incoming job stream to the node specified by nodename. JOBB is a valid JOB statement that allows the subsequent job stream entry into the destination node for execution.

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

MVS/System Product - JES2 incorporates the function previously provided by Network Job Entry Facility for JES2 (JES2 NJE) Program Number 5740(XR8).

Therefore, pages 329 through 348 may be removed from the JCL manual. However, if your installation is to transmit jobs to a JES2 NJE installation, retain these pages as a reference for programmers who are preparing jobs for transmission and execution on the JES2 NJE node.

The JOBPARM Statement

Control Statement

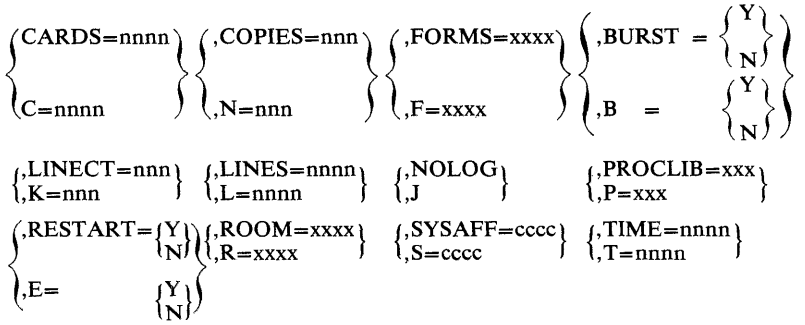
The JOBPARM statement specifies job related parameters for JES2 NJE.

The JOBPARM statement consists of the characters /* in columns 1 and 2, the word JOBPARM in columns 3-9, a blank in column 10, and parameters in columns 11-71. Columns 72-80 are ignored.

For further information, see "Obtaining Output (JES2)."

```
/*JOBPARM parameters
```

Code one or more of the following parameters in the longer form (full word) or the shorter form (one letter abreviation).



CARDS=nnnn
a value estimating the number of output cards from this job (from 0 to 9999999 cards).

COPIES=nnn
a value indicating the number of printed output copies of a job related output that is to be produced (from 1 to 255 copies). The upper limit of this value can be reduced during JES2 NJE initialization.

FORMS=xxxx
an alphameric value indicating the print and punch forms for this job's output that are not further defined in this job (from 1 to 4 characters).

BURST=Y or N
indicates the default burst characteristic of all SYSOUT data sets produced by this job. This characteristic is used to either burst the data set (Y), or to continuously fold the data set (N), when the data set is directed to an IBM 3800 printer with the burst option installed.

LINECT=nnn
a value showing the number of lines to put on each output page for JES2 NJE page overflow processing (from 0 to 255 lines).

LINES=nnnn
a value estimating the number of output lines from this job — in thousands of lines (from 0 to 9999).

NOLOG
a parameter meaning that you do not want the JES2 NJE job log as output. (The job log contains a list of job related console messages and operator replies produced during processing of the job.)

PROCLIB=xxx
an alphameric value indicating the DDNAME of the cataloged procedure library that is to be used to convert the JCL for this job. (This name refers to a DD statement in the JES2 NJE cataloged procedure.)

RESTART
if this job is executing before a re-IPL and JES2 NJE warm start and cannot be restarted from a step or checkpoint, JES2 NJE does one of the following:

- Y indicates that the job is queued for re-execution from the beginning of the job.
- N indicates that no special action is to be taken.

If RESTART is not specified, N is assumed. However, the installation may override this default by JES2 NJE initialization parameters.

ROOM=xxxx
an alphameric value indicating a programmer's room number to be placed on the job's separators for routing SYSOUT data sets back to the programmer (from 1 to 4 characters).

SYSAFF=cccc
1 to 7 system affinities can be specified indicating systems to be eligible to process this job. In order to specify more than one system, code: SYSAFF=(cccc,cccc,...). cccc is an alphameric value indicating one or more of the following:

- * indicates the system into which the job was read.
- ANY indicates any system in the JES2 NJE multi-access spool configuration.
- cccc indicates a specific system. "cccc" must be the four alphameric character system-id of one of the systems in the JES2 NJE multi-access spool configuration.
- ,IND when included after any of the above specifications, indicates systems scheduling in independent mode.

TIME=nnnn
a value estimating the job execution time in minutes of real time (from 0 to 279,620 minutes).

Rules for Coding

- Any JOBPARM statement parameter value will supersede the equivalent parameter value from the accounting field (in HASP format) of the JOB statement or from any preceding JOBPARM statement in this job's JCL. All of these statements override the default established by the installation.
- Any number of the above parameters may be placed on a single JOBPARM statement and as many JOBPARM statements as desired may be placed together with a given input stream. The JOBPARM statement cannot be continued.
- Place the JOBPARM statement after the JOB statement.
- If you code the PROCLIB parameter on the JOBPARM statement, the name of the DD statement should be in the JES2 NJE cataloged procedure. If it is not, the JES2 NJE default procedure is used.
- If you code LINECT=0, JES2 NJE will not eject to a new page when the number of lines has exceeded the page limit that was established at JES2 NJE initialization.
- The COPIES parameter does not apply when FREE= is specified on the SYSOUT DD statement.
- If you code COPIES=000, JES2 NJE will default the copy count to one.
- The JOBPARM statements will be ignored on any node other than the node of execution. However, if the JOBPARM card is placed before the ROUTE card, it will be processed at both the input node and the execution node.

Example of the JOBPARM Statement

```
/*JOBPARM    L=60,R=4222,T=50
```

The three specifications mean the following:

L=60	The job's estimated printed output will be 60,000 lines.
R=4222	The programmer's room is 4222. This information will be placed in the separators for both printed and punched data sets.
T=50	The job's estimated execution time is 50 minutes.

The MESSAGE Statement

Control Statement

The MESSAGE statement permits you to send messages to the operator (via the operator console) at JES2 NJE job input time.

The MESSAGE statement consists of the characters /* in columns 1 and 2, the word MESSAGE in columns 3-9, and at least one blank followed by the message. Columns 72-80 are ignored.

```
/*MESSAGE   message to be written
```

Rules for Placement

- Place the MESSAGE statement after the JOB statement to allow the job number to be appended to the beginning of the message.
- If the MESSAGE statement is not included within the boundaries of a job, the input device name is appended to the beginning of the message.
- The message will only be sent to the operator on the execution node unless the MESSAGE card is placed before the ROUTE card. The latter will cause the message to be printed on both the input and execution nodes.

Example of the MESSAGE Statement

```
/*MESSAGE CALL DEPT58 WHEN PAYROLL JOB IS FINISHED--EX.1946
```

Requests that the operator call department 58 when the payroll job is complete.

The NETACCT Statement

The NETACCT statement specifies an account number which is available to all the nodes in a network; the account number may be used as is or may be translated to local account numbers as desired. 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. Columns 72-80 are ignored.

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

network-account-number
From 1 to 8 alphameric characters representing this job's accounting through the network.

Rules for Coding

Observe the following rules for coding the NETACCT statement:

- Place the NETACCT statement after the JOB statement.
- If more than one NETACCT card is included, the last network account number is used.
- The NETACCT card is ignored on any node other than the input node.
- If a NETACCT statement is not supplied, JES2 NJE attempts to find a network account number using the local account number in a table lookup.
- If both NETACCT and a local account number are supplied, the local account number is used on the input node.

Example of the NETACCT Statement

```
/*NETACCT NETNUM10
```

The network account number, NETNUM10, will be transmitted with the job to various nodes, available for local interpretation at each node.

JES2 NJE

MESSAGE NETACCT

The NOTIFY Statement

Control Statement

The NOTIFY statement causes a job's notification messages to be directed to the userid specified on the NOTIFY statement. Print, punch, and execution routings are not affected. Use of the NOTIFY statement overrides any NOTIFY specification on the JOB statement.

The NOTIFY statement consists of the characters /* in columns 1 and 2, the word NOTIFY in columns 3-8, and one or more blanks followed by a parameter. Columns 72-80 are ignored.

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

nodename.userid
a node name and user identifier (each from 1 to 8 characters) separated by a delimiter. The delimiter may be a period (.) as shown, a colon (:), a slash (/), or paired parentheses (enclosing the userid). nodename must consist of alphameric characters. userid may contain any valid characters except the comma (,), right parenthesis ()), or blank character. Specification of nodename.userid causes the origin node field in the job's NJE job header to be set to the node specified on the /*NOTIFY statement even though the actual job origin node may be different.

userid
a user identifier of 1 to 8 alphameric characters. Specification of userid causes notification messages to be sent to the job's origin node. (The origin node name is not overridden by the /*NOTIFY statement.)

Examples of the NOTIFY Statement

```
/*NOTIFY VMNODE.VMUSER
```

Notification messages are sent to user VMUSER on node VMNODE.

```
/*NOTIFY TSOUSER
```

Notification messages are sent to user TSOUSER on the job's origin node.

The OUTPUT Statement

Control Statement

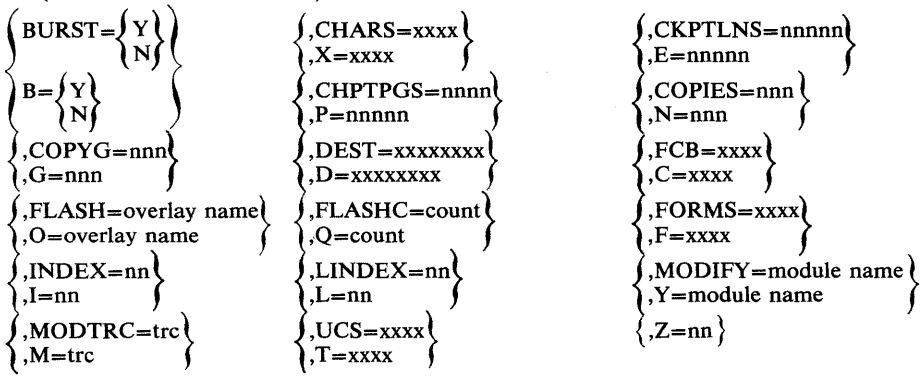
The OUTPUT statement specifies characteristics and/or options of a specific SYSOUT data set or group of SYSOUT data sets.

For further information on the OUTPUT statement, see "Obtaining Output (JES2)".

The OUTPUT statement consists of the characters /* in columns 1 and 2, the word OUTPUT in columns 3-8, and a code beginning in column 10 followed by a blank and the keyword parameters. Columns 72-80 are ignored.

```
/*OUTPUT    code    parameters
```

Code one or more of the following parameters in the longer form (full word) or the shorter form (one letter abbreviation).



code
alphameric characters referring to all SYSOUT data sets within your job whose code in the form number subparameter of the SYSOUT parameter matches the "code" specified on the OUTPUT statement (from 1 to 4 characters). Specifying code as "*" indicates that this OUTPUT statement is a continuation of the previous OUTPUT statement.

BURST
Y indicates that the printed output from a 3800 printer is to be burst into separate sheets. N indicates that the printed output is to be in continuous, fanfold mode. N is the default.

CHARS=xxxx
the name of a character arrangement table for a 3800 printer. Each name is 1 to 4 alphameric or national characters; from one to four names can be coded. To specify more than one name, code: CHARS=(xxxx,xxxx...).

CKPTLNS=nnnnn
a value (from 0 to 32767 for printers and 1 to 32767 for punches) that specifies the maximum number of lines or cards contained in a logical page. The default is the value specified in the JES2 initialization parameter for the device.

CKPTPGS=nnnnn
a value from 1 to 32767 that specifies the number of logical pages to be printed before the next checkpoint is taken. The default is the value specified in the JES2 initialization parameter for the device.

COPIES=nnn
a value indicating the number of copies of printed job-related output that is to be produced (from 1 to 255 copies). If you code COPIES=000, JES2 NJE will default the copy count to 1.

COPYG=nnn

a value that specifies how many copies of each page of the printed output are to be grouped together. When COPYG is specified, it determines the number of copies to be printed (that is, if COPYG is coded, COPIES is ignored). The total number of copies printed equals the sum of the specified group values. The sum of the group values cannot be greater than 255. To specify more than one group value, code: COPYG=(nnn,nnn...). The maximum number of groups that can be specified is 8.

Note: This parameter applies only to the IBM 3800 Printing Subsystem (a nonimpact printer). If COPYG is coded and an impact printer is used, COPYG is ignored.

DEST=xxxxxxxx

One to four different destinations can be specified for each output data set. To specify more than one destination, code: DEST=(xxx,xxx...).xxx is an alphameric value indicating one of the following:

- Rnnn, RMnnn, or RMTnnn - remote terminal, nnn indicating a 1- to 3-digit numeric value specifying the remote terminal number.

Note: R0 is equivalent to LOCAL.

- Unnn - local terminal, nnn indicating a 1to 3-digit decimal value specifying the local device with special routing.
- LOCAL - any local device.
- name - a 1- to 8-character (alphameric or national) name of a remote or local device (as defined by the system programmer).
- Nnn - a node identified by nn which is a 1-2 digit decimal number between 1 and 99 inclusive.
- NnnRmmm - a node identified by nn which is a 1-2 digit decimal number between 1 and 99 inclusive and a remote workstation that is connected to the node and identified by mmm which is a 1-3 digit decimal number between 1 and 255 inclusive. A specification of RO is equivalent to the specification LOCAL at mode nn.
- nodename.userid - a node name and user or remote identifier (each from 1 to 8 characters) separated by a delimiter. The delimiter may be a period (.) as shown, a colon (:), a slash (/), or paired parentheses (enclosing the userid). Nodename must consist of alphameric characters. Userid may contain any valid characters except the comma (,), right parenthesis ()), R, RM, RMT, or blank character. This parameter is used to route information between JES2 NJE nodes and virtual machines at VM network job interface (NJI) nodes or remote work stations at ASP NJI nodes.

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

FCB=xxxx

an alphameric value indicating the data set forms control or carriage specifications (from 1 to 4 characters).

FLASH=overlay name

the name (1 to 4 alphameric or national characters) of the forms overlay frame that the operator is to insert into the 3800 printer before printing begins.

FLASHC=count

a value, between 0 and 255, that indicates the number of copies to be flashed with the overlay, beginning with the first copy printed. For the 3800 printer, if FLASH is specified and FLASHC is omitted, all copies are flashed.

FORMS=xxxx

an alphameric value indicating the print and punch forms (from 1 to 4 characters).

INDEX=nn
a value indicating the data set indexing print position offset (to the right) for the 3211 printer (from 1 to 31).

LINDEX=nn
a value indicating the data set indexing print position offset (to the left) for the 3211 printer (from 1 to 31).

MODIFY=module name
the name (1 to 4 alphameric or national characters) of a copy modification module previously stored in SYS1.IMAGELIB that is used to replace variable data in the printed data set of the 3800 printer.

MODTRC=trc
the table reference character (0-3) that identifies a character arrangement table specified on the CHARS parameter.

UCS=xxxx
an alphameric value indicating the universal character set specification (from 1 to 4 characters).

Z=nn
a numeric value (0-99) indicating a compaction table, identified by nn, is to be used when sending the SYSOUT described by this control statement to a SNA remote terminal. This specification overrides any remote-device-defined compaction table value. Setting Z to zero turns off the compaction feature. When no value is specified, the remote-device-defined compaction table is used.

Note: If you have Release 3 of JES2 NJE installed, the Z parameter has no effect on compaction for SNA application-to-application sessions; such compaction is done only on a session basis.

General Rules for Coding

- Specifying code as "*" causes continuation of the previous OUTPUT statement, regardless of the position of the previous statement (that is, the previous statement does not have to immediately precede the continuation). The first OUTPUT statement cannot specify code as "*"
- Parameters specified on the OUTPUT statement will replace any equivalent parameters specified on the referenced DD statement.
- Code as many OUTPUT statements as you need. If more than one OUTPUT statement has the same "code" starting in column 10, the first OUTPUT statement parameters are used. If there are duplicate parameters on the same OUTPUT statement, the last parameter overrides any preceding duplicate parameter (except on the DEST parameter).
- Use the shorter form of the parameters when coding several parameters.
- Place the OUTPUT statement after the JOB statement.
- OUTPUT statements placed after a /*ROUTE XEQ statement are ignored except at the execution node.
- OUTPUT statements placed before a /*ROUTE XEQ statement are processed on both the input and execution nodes.

Rules for Coding DEST

- If more than one destination is coded, the destinations must be in parentheses. If only one destination is coded, the parentheses are optional.
- When using the nodename.userid form of the DEST parameter, you may use continuation statements to specify up to 4 destinations. The continuation statement must contain the characters /* in columns 1 and 2, OUTPUT in columns 3 to 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      F=STD.,T=P11,C=8LPI,DEST=(FIRST.DEST)
/*OUTPUT     *         DEST=(SECOND.USER,THIRD.REMOTE)
```

Rules for Coding FCB

- If the printer on which the data set is to be printed does not have the forms control buffer feature, the operator is sent a message to mount the proper carriage control tape.
- Do not specify STD1 or STD2 unless the installation indicates that you should.

Rules for Coding INDEX and LINDEX

If the 3211 printer has the INDEX feature, it will offset the first physical print position to the right by the number of print positions specified to cause the total print line width to be reduced by the number of print positions specified. (That is, a specification of 30 will mean that the maximum line width is now 30 positions less than the original value.) These parameters are ignored on printers other than the 3211.

Example of the OUTPUT Statement

```
/*OUTPUT ABCD COPIES=6,COPYG=(1,2,3),DEST=RMT23
```

Refers to all SYSOUT data sets within the job whose DD statement specified 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 PRIORITY Statement

Control Statement

The PRIORITY statement assigns a queue selection priority to a job. This selection priority applies to all of the queues (except hardcopy) to which the job or its output might be queued. A priority value specified on a PRIORITY statement overrides any priority specified with the PRTY parameter on a JOB statement.

Note: Depending on the JES2 NJE initialization options specified, the PRIORITY statement might be ignored. In a JES2 NJE network, the PRIORITY statement is ignored on any node other than the node from which it was input.

For further information on the use of PRIORITY, see "Routing a Job Through the System (JES2)".

The PRIORITY statement consists of the characters /* in columns 1 and 2, the word PRIORITY in columns 3-10, and at least one blank followed by the priority specification. Columns 72-80 are ignored.

```
/*PRIORITY  p
```

p
a number between 0 and 15 that indicates the priority of the job.

Default: If PRIORITY is not present, or if PRIORITY is ignored, priority is derived using information from (1) the PRTY parameter on the JOB statement, (2) the accounting information on the JOBPARM statement, (3) the accounting information on the JOB statement, or (4) an installation-defined default.

Rule for Coding

- The PRIORITY statement must immediately precede the JOB statement. If it does not, or if "p" is not a number between 0 and 15, the PRIORITY statement is ignored and the input stream is flushed until a JOB statement or another PRIORITY statement is found.

Example of the PRIORITY Statement

```
/*PRIORITY  7
```

The job has a queue selection priority of 7. This value only has meaning in relation to other jobs in the system.

JES2 NJE

PRIORITY

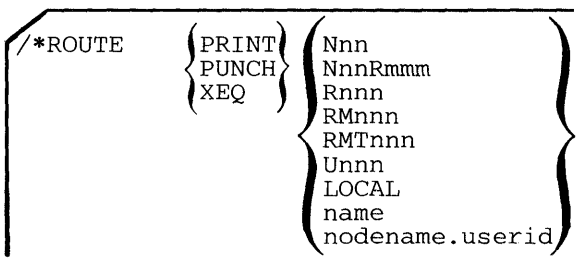
The ROUTE Statement

Control Statement

The ROUTE statement specifies the destination of the output which is not specifically routed using the DEST parameter.

The ROUTE statement consists of the characters /* in columns 1 and 2, ROUTE in columns 3-7, at least one blank followed by PRINT, PUNCH, or XEQ, then at least one blank followed by one of the device specifications. The device specification must be followed by at least one blank, which must precede column 72. Columns 72-80 are ignored.

For further information, see "Obtaining Output (JES2)."



PRINT
specifies that the job's printed output is to be routed.

PUNCH
specifies that the job's punched output is to be routed.

XEQ
specifies that the job is to be routed for execution.

Nnn
a node identified by nn which is a 1-2 digit decimal number between 1 and 99 inclusive.

NnnRmmm
a node identified by nn which is a 1-2 digit decimal number between 1 and 99 inclusive, and a remote workstation that is connected to the node and identified by mmm which is a 1-3 digit decimal number between 1 and 255 inclusive. Specification of R0 is equivalent to the specification LOCAL at node nn.

Rnnn

RMnnn

RMTnnn
remote terminal, nnn indicating a 1- to 3-digit numeric value defining the remote terminal to which the output is to be sent. The valid range of nnn is 1 to the maximum number of work stations assigned to JES2. The number of work stations cannot exceed 255.

Note: R0 is equivalent to LOCAL.

Unnn
local terminal, nnn being a 1- to 3-digit decimal value that specifies a specific local device to which the output is to be sent. The valid range for nnn is from 1 to 255, inclusive.

LOCAL
any local device at the originating node.

name
a 1- to 8-character (alphameric or national) name of a remote or local device (as defined by the system programmer) that is to receive the output.

nodename.userid
a node name and user or remote identifier (each from 1 to 8 characters) separated by a delimiter. The delimiter may be a period (.) as shown, a colon (:), a slash (/), or a paired parentheses (enclosing the userid). Nodename must consist of alphameric characters. Userid may contain any valid characters except the comma (,) right parenthesis ()), R, RM, RMT, or blank character. This parameter is used to route information between JES2 NJE nodes and virtual machines at VM network job interface (NJI) nodes or remote work stations at ASP NJI nodes.

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

Rules for Coding

- Place the ROUTE statement after the JOB statement.
- A ROUTE statement can be used to direct job execution or print or punch routing of output. However, each ROUTE statement can only contain one routing. Therefore, if execution, print, and punch are all to be routed, three statements must be used.
- If more than one ROUTE statement has the same category (PRINT, PUNCH, or XEQ), the last ROUTE statement for each category will be used.
- /*ROUTE XEQ statements are only processed on the input node and should precede any //SYSIN DD statement.
- /*ROUTE PRINT or /*ROUTE PUNCH statements following a /*ROUTE XEQ statement are only processed on the execution node. A /*ROUTE PRINT or /*ROUTE PUNCH statement in front of a /*ROUTE XEQ statement is processed on both input and execution nodes.

Examples of the ROUTE Statement

```
/*ROUTE      PRINT    RMT6
```

Routes printed output to remote terminal 6.

```
/*ROUTE      PUNCH    PUNCH2
```

Routes punched output to device "PUNCH2" as defined by the system programmer.

```
/*ROUTE XEQ DENVER
```

Routes job to node named DENVER for execution.

The SETUP Statement

Control Statement

SETUP is a control statement which is used to indicate volumes needed for executing a phase of the job.

Use of the SETUP statement in a JES2 NJE network generally requires that the SETUP statement follow any ROUTE XEQ statement. This will prevent the setup from being requested on a node other than the node of execution. If the SETUP statement is processed prior to processing of the ROUTE or XEQ statements, setup will be requested on both the input and execution nodes.

For further information on the use of the SETUP statement, see "Routing a Job Through the System (JES2)".

The SETUP statement consists of the characters /* in columns 1 and 2, the word SETUP in columns 3-7, and at least one blank followed by the volume serial numbers. Columns 72-80 are ignored.

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

volume serial number
identifies the volume or volumes required for execution of the job.

Rules for Coding

- All SETUP statements should be placed after the JOB statement.
- As many SETUP statements as necessary can be used.

Example of the SETUP Statement

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

The two volumes requested are listed on the console when the job enters the system. The job is then placed in the hold status awaiting release by the operator when the required volumes are available. The message informs the operator that the volumes should be mounted before the job is run.

The SIGNOFF Statement

Control Statement

SIGNOFF is a control statement that indicates to the central computer that the user wishes to terminate a remote job stream processing session. At the completion of the current print and/or punch streams, JES2 NJE disconnects the station from the system. If jobs are being read into the system from the remote station when the output is completed, JES2 NJE disconnects the remote station when the input is completed.

Both SNA (systems network architecture) and BSC (binary synchronous communication) remote work stations can use the SIGNOFF statement. SNA remote stations, however, can also use the LOGOFF command to end a session with JES2 NJE. The LOGOFF command has some options that are not provided by the SIGNOFF statement. For a discussion of the LOGOFF command, refer to **System Programming Library: Network Job Entry Facility for JES2** and **OS/VS2 MVS System Programming Library: VTAM**.

The SIGNOFF statement consists of the characters /* in columns 1 and 2, and the word SIGNOFF in columns 3-9.

```
/*SIGNOFF
```

Example of the SIGNOFF Statement

```
/*SIGNOFF
```

Requests that a remote job stream processing session be terminated.

JES2 NJE

SETUP
SIGNOFF

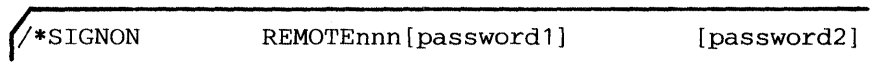
The SIGNON Statement

Control Statement

SIGNON is a control statement that indicates to the central computer that the user wishes to begin a remote job stream processing session. The SIGNON statement overrides the remote identification number normally assigned to the remote station. This statement is optional for all work stations except non-multi-leaving remote stations on a switched line.

Note: SNA remote work stations must use the LOGON command instead of the SIGNON statement to notify JES2 NJE of a connection request. For a discussion of the LOGON command, refer to **System Programming Library: Network Job Entry Facility for JES2** and **OS/VS2 MVS System Programming Library: VTAM.**

The SIGNON statement consists of the characters /* in columns 1 and 2, the word SIGNON in columns 3-8, and REMOTEnnn starting in column 16. The SIGNON statement can also include two passwords: one beginning in column 25 and the other in column 73.



```
/*SIGNON         REMOTEnnn[password1]          [password2]
```

nnn
a value specifying the remote identification number assigned to the remote station that is requesting to sign on.

password1
a password, assigned to a dial line, that allows the remote station access to JES2 NJE for remote job stream processing. This password is established at JES2 NJE initialization, and can be changed or deleted by the operator with the $T command.

password2
a password that ensures that the remote station signing on is a valid RJE (remote job entry) station. This password is established at JES2 NJE initialization.

Rules for Coding

- For multi-leaving remote stations, the SIGNON statement is put at the end of the JES2/RTP program deck.
- For non-multi-leaving remote stations, the SIGNON statement is transmitted alone as part of the initial connection process.
- Code REMOTEnnn with the same character string as the RMTnnn parameter of the ROUTE statement.

Example of the SIGNON Statement

```
/*SIGNON         REMOTE123PSWD
```

Requests that remote station 123 begin a remote job stream processing session. PSWD, beginning in column 25, is the password assigned to a dial line.

The XEQ Statement

The XEQ statement routes a job to a node for execution. It performs the same function as /*ROUTE XEQ (see "The ROUTE Statement" earlier in this chapter). The XEQ statement consists of the characters /* in columns 1 and 2, XEQ in columns 3-5, and a 1-8 character destination node name starting in column 10.

```
/*XEQ  {name}
      {Nnn }
```

name
Specifies a 1-8 alphameric or special-character name (specified at initialization) of the node which is to receive the job for execution.

Nnn
Specifies the number (nn) of the node in the network which is to receive a job for execution.

Rules for Coding

Observe the following rules for coding the XEQ statement:

- Place the XEQ statement after the job statement.
- If more than one XEQ statement is included within the job, the last XEQ statement's node name is used.
- The XEQ statement is ignored on any node other than the input node.

Example

```
/*XEQ   ATLANTA
```

The job is to be routed to and executed on the node defined as ATLANTA. The output, by default, returns to the node of origin.

JES2
NJE

SIGNON
XEQ

Reference Tables

The first section of this appendix summarizes the DD statement parameters required to perform the following functions:

figure 19

- Create a data set on an unit record device (card punch or printer)
- Create a data set on a system output device
- Create a data set on magnetic tape
- Create a data set on a direct access device
- Create a data set on Mass Storage System
- Create a subsystem data set

figure 20

- Retrieve a data set fron an unit record device (card reader or paper tape reader)
- Retrieve a data set from the input stream
- Retrieve an associated data set
- Retrieve a passed data set (magnetic tape or direct access)
- Retrieve a cataloged data set (magnetic tape or direct access)
- Retrieve a kept data set (magnetic tape or direct access)

figure 21

- Extend a passed data set (magnetic tape or direct access)
- Extend a cataloged data set (magnetic tape or direct access)
- Extend a kept data set (magnetic tape or direct access)

Also included are tables for:

- retrieving or extending an indexed sequential data set
- area arrangement of indexed sequential data sets
- mutually-exclusive DD parameters
- disposition processing
- direct access capacities
- track capacities
- allowable DSPs for PROCESS statements
- the JOB statement
- the EXEC statement
- the DD statement

REF TAB

Device	Parameter Type	Parameter	Comments
Unit Record Devices	Location of the Data Set	UNIT	Required
	Data Attributes	DCB	Optional
	Special Processing Options	UCS	Optional (for a printer with the universal character set feature)
		FCB	Optional (for a 3211 or 3800 printer if forms control information is to be specified)
		FREE	Optional
		DUMMY	Optional
		COPIES	Optional
		CHARS	Optional (for the 3800 Printing Subsystem)
		BURST	Optional (for the 3800 Printing Subsystem)
		FLASH	Optional (for the 3800 Printing Subsystem)
		MODIFY	Optional (for the 3800 Printing Subsystem)
System Output Devices	Location of the Data Set	SYSOUT	Required. Specifies the output class
	Data Attributes	DCB	Optional
	Special Processing Options	OUTLIM	Optional
		FREE	Optional
		DEST	Optional
		DSID	Required for output to a 3540 diskette
		HOLD	Optional
		UCS	Optional (for a printer with the universal character set feature)
		FCB	Optional (for a 3211 or 3800 printer if forms control information is to be specified)
		COPIES	Optional
		CHARS	Optional (for the 3800 Printing Subsystem)
		BURST	Optional (for the 3800 Printing Subsystem)
		FLASH	Optional (for the 3800 Printing Subsystem)
		MODIFY	Optional (for the 3800 Printing Subsystem)
Magnetic Tape	Data Set Information	DSNAME (or DSN)	Required if the data set is to be cataloged or used by a later job
		DISP	Required if the data set is to be cataloged, used by a later step in this job, or used by another job
	Location of the Data Set	UNIT	Required unless you request (with the VOLUME parameter) the same volume used for an earlier data set in your job
		VOLUME (or VOL)	Required if you want a specific volume. If you do not use this parameter you will get a scratch tape
		LABEL	Required if you do not want to use IBM standard labels for the data set
	Data Attributes	DCB	Optional
	Special Processing Options	DUMMY	Optional
		CHKPT	Optional
		FREE	Optional
		PROTECT	Optional

Figure 19. DD Parameters for Creating a Data Set (Part 1 of 2)

Device	Parameter Type	Parameter	Comments
Direct Access Devices	Data Set Information	DSNAME (or DSN)	Required if the data set is to be cataloged or used by a later job
		DISP	Required if the data set is to be cataloged, used by a later step in this job, or used by another job
	Location of the Data Set	UNIT	Required unless you request (with the VOLUME parameter) the same volume used for an earlier data set in your job
		VOLUME (or VOL)	Required if you want a specific volume or multiple volumes. If you do not use this parameter your data set will be allocated on any suitable volume
		LABEL	Required if you want the data set to have both IBM standard and user labels
	Size of the Data Set	SPACE	SPACE must be used for ISAM data sets
	Special Processing Options	DUMMY	Optional
		DYNAM	Optional
		FREE	Optional
		PROTECT	Optional
Mass Storage System (MSS)	Data Set Information	DSNAME (or DSN)	Required if the data set is to be cataloged or used by another job
		DISP	Required if the data set is to be cataloged, used by a later step in the job, or used by another job
	Location of the Data Set	UNIT	Required unless you request (with the VOLUME parameter) the same volume used for an earlier data set in your job
		VOLUME (or VOL)	Required for specific volume requests. Use MSVGP instead of VOL=SER if a nonspecific volume in a specific MSS volume group is desired. If neither is coded, the system will select an already mounted 3330V volume (storage or public) unless PRIVATE is coded
		LABEL	Required if you want the data set to have both IBM standard and user labels
		MSVGP	Required if a nonspecific volume in a specific MSS volume group is required
	Size of the Data Set	SPACE	Required unless MSVGP is coded
	Special Processing Options	DUMMY	Optional
		DYNAM	Optional
		FREE	Optional
		PROTECT	Optional
Subsystem Data Set	Location of the Data Set	SUBSYS	Required. Specifies the subsystem and the subsystem-defined parameters
	Data Attributes	DCB	Optional
	Special Processing Options	DUMMY	Optional
		FREE	Optional

Figure 19. DD Parameters for Creating a Data Set (Part 2 of 2)

Data Set	Parameter Type	Parameter	Comments
Unit Record Devices	Location of the Data Set	UNIT	Required
	Data Attributes	DCB	Optional
	Special Processing Options	DUMMY	Optional
		FREE	Optional
Input Stream	Location of the Data Set	*	You must specify **one** of these parameters
		DATA	
	Special Processing Options	DLM	Optional
		FREE	Optional
Associated Data Set	Location of the Data Set	*	You must specify **one** of these parameters
		DATA	
	Data Set Information	DSID	Required for 3540 associated data sets
		VOL=SER	Optional for 3540 associated data sets
Passed Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required only if you want more units
		LABEL	Required only if the data set does not have IBM standard labels
	Data Attributes	DCB	Optional
	Special Processing Options	FREE	Optional
		CHKPT	Optional
		DUMMY	Optional
Cataloged Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Optional
		VOLUME	May be required if you want to begin processing with a volume after the first
		LABEL	Required only if the data set does not have IBM standard labels
	Special Processing Options	DUMMY	Optional
		DYNAM	Optional
		FREE	Optional
		CHKPT	Optional
Kept Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required
		VOLUME	Required
		LABEL	Required only if the data set does not have IBM standard labels
	Data Attributes	DCB	Optional
	Special Processing Options	DUMMY	Optional
		DYNAM	Optional
		FREE	Optional
		CHKPT	Optional

Figure 20. DD Parameters for Retrieving a Data Set

Data Set	Parameter Type	Parameter	Comments
Passed Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required only if you want more units
		VOLUME	Required only if you need more volumes
		LABEL	Required only if the data set does not have IBM standard labels
	Size of the Data Set	SPACE	Required only if you want to override the secondary quantity
	Data Attributes	DCB	May be required if data set does not have IBM standard labels
	Special Processing Options	FREE	Optional
		CHKPT	Optional
		DUMMY	Optional
Cataloged Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Optional
		VOLUME	Required only if you need more volumes
		LABEL	Required only if the data set does not have IBM standard labels
	Size of the Data Set	SPACE	Required only if you want to override the secondary quantity
	Data Attributes	DCB	Required only if the data set does not have IBM standard labels
	Special Processing Options	DUMMY	Optional
		FREE	Optional
		CHKPT	Optional
Kept Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required
		VOLUME	Required
		LABEL	Required only if data set does not have IBM standard labels
	Size of the Data Set	SPACE	Required only if you want to override the secondary quantity
	Data Attributes	DCB	Required only if the data set does not have IBM standard labels
	Special Processing Options	DUMMY	Optional
		DYNAM	Optional
		FREE	Optional
		CHKPT	Optional

Figure 21. DD Parameters for Extending a Data Set

Area	Parameter Type	Parameter	Comments
Index (used only if index area not on same device type as prime area) (First DD statement)	Data Set Information	DSNAME	Required. You must code the same value as in second DD statement.
		DISP	Required. You must code the same value as in second DD statement.
	Location of the data set	UNIT	Required
		VOLUME	Required
	Data Attributes	DCB	Required
Prime and Overflow; or Index, Prime, and Overflow; or Index and Prime (required) (Second DD statement)	Data Set Information	DSNAME	Required
		DISP	Required. Specifies whether you are retrieving the data set.
	Location of the data set	UNIT	Required unless it is a passed data set with all three areas on one volume.
		VOLUME	Same requirement as UNIT. If used, code volumes in order they were defined.
	Data Attributes	DCB	Required
Overflow (used only if overflow area not on same device type as prime area) (Third DD Statement)	Data Set Information	DSNAME	Required. You must code the same value as in second DD statement.
		DISP	Required. You must code the same value as in the second DD statement.
	Location of the data set	UNIT	Required
		VOLUME	Required
	Data Attributes	DCB	Required

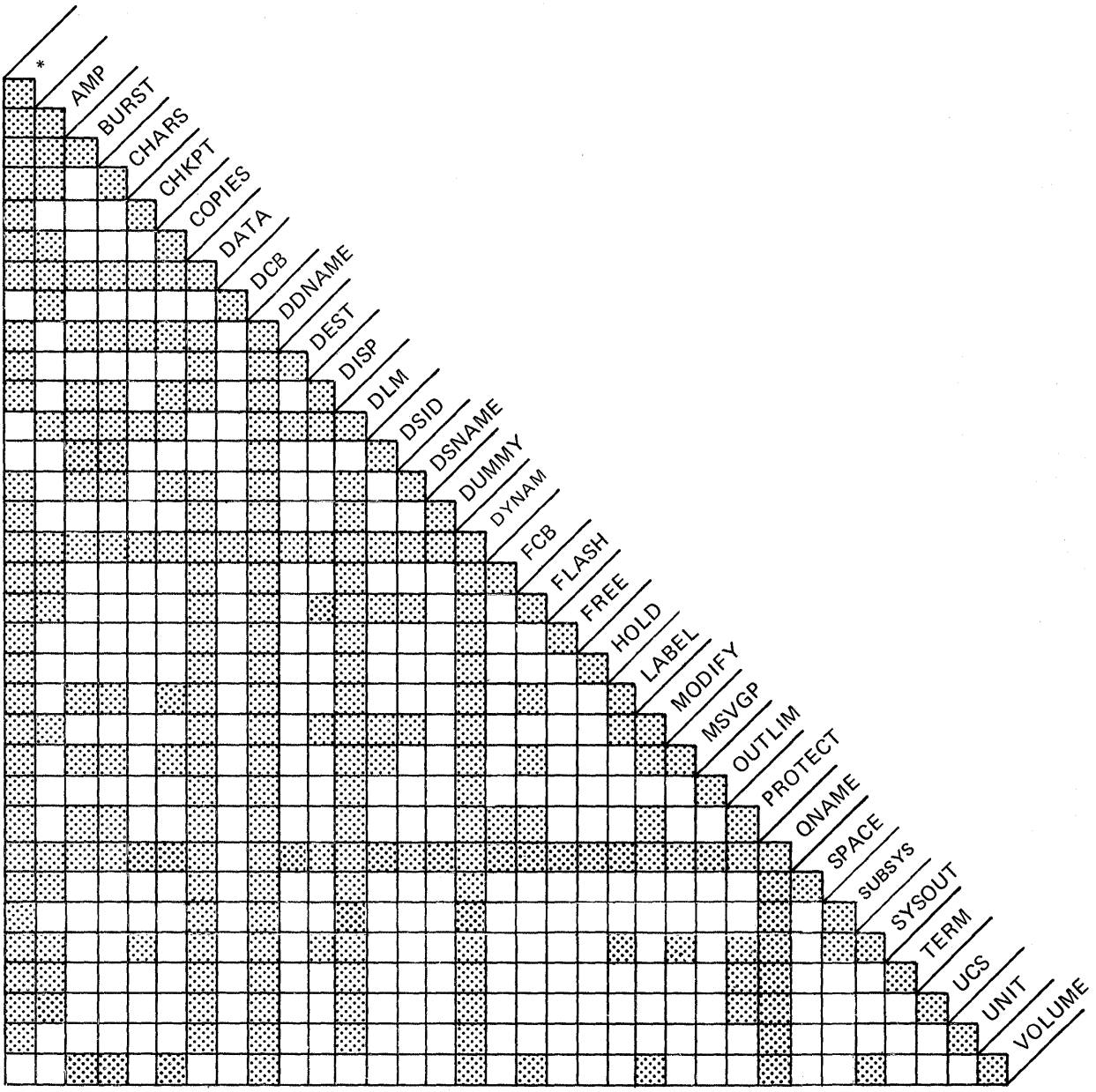
Figure 22. DD Parameters for Retrieving or Extending an Indexed Squential Data Set

CRITERIA			RESTRICTIONS ON DEVICE TYPES AND NUMBER OF DEVICES REQUESTED	RESULTING ARRANGEMENT OF AREAS
1. Number of DD statements	2. Area defined on a DD statement	3. Index size coded ?		
3	INDEX PRIME OVFLOW	-	None	Separate index, prime, and overflow areas.
2	INDEX PRIME	-	None	Separate index and prime areas.[1]
2	PRIME OVFLOW	No	None	Separate prime and overflow areas. An index area is at the end of the overflow area.
2	PRIME OVFLOW	Yes	The statement defining the prime area cannot request more than one device.	Separate prime and overflow areas. An index area is embedded in the prime area.
1	PRIME	No	None	Prime area with index area at its end.[2]
1	PRIME	Yes	Cannot request more than one device.	Prime area with embedded index area.[2]

[1] If both areas are on volumes that correspond to the same device type, an overflow area is established if one of the cylinders allocated for the index area is only partially used. The overflow area is established in the unused portion of that cylinder.

[2] If the index area is at least one cylinder and the unused portion of the index area is less than one cylinder, the unused portion is used as an overflow area. For a one-cylinder data set, the index is not assigned a whole cylinder; therefore, no overflow area is provided.

Figure 23. Area Arrangement of Indexed Sequential Data Sets



Legend: This table shows which DD parameters cannot be coded together. At the intersection of the horizontal and vertical columns, the square will be shaded if the parameters are mutually exclusive and white if they can be coded together on the same DD statement.

For example, to see if DISP and SYSOUT are mutually exclusive, look down the column marked DISP and across the column marked SYSOUT. In this case, they are mutually exclusive.

As indicated by the table, each DD parameter is mutually exclusive with itself; that is, it cannot appear twice on the same DD statement.

Figure 24. Table of Mutually Exclusive DD Parameters

Status	Requested Disposition	Conditional Disposition	Action Taken at Normal End of Step[1]	Action Taken at Abnormal End of Step[1], when Step Fails Due to:		Action Taken at End of Job
				Program canceled or abnormally terminated	Subsequent data set allocation for same step failed	
NEW or MOD[2]	none	none	deleted	deleted	deleted	
	KEEP	none	kept	kept	deleted	
	DELETE	none	deleted	deleted	deleted	
	CATLG	none	cataloged	cataloged	deleted	
	PASS	none	passed	passed	passed	deleted
	PASS	any	passed	passed	passed	deleted[5]
	any except PASS	KEEP	requested disposition	kept	deleted	
	any except PASS	DELETE	requested disposition	deleted	deleted	
	any except PASS	CATLG	requested disposition	cataloged	deleted	
OLD or MOD or SHR	none	none	kept	kept	kept	
	KEEP	none	kept	kept	kept	
	DELETE	none	deleted	deleted	kept	
	CATLG	none	cataloged[3]	cataloged[3]	kept	
	UNCATLG	none	uncataloged	uncataloged	kept	
	PASS	none	passed	passed	passed	kept
	PASS	any	passed	passed	passed	kept[6]
	any except PASS	KEEP	requested disposition	kept	kept[4]	
	any except PASS	DELETE	requested disposition	deleted	kept[4]	
	any except PASS	CATLG	requested disposition	cataloged[3]	kept[4]	
	any except PASS	UNCATLG	requested disposition	uncataloged	kept[4]	

Footnotes:

[1] See list of exceptions in right-hand column.

[2] If volume information is not available to the system, a MOD data set is considered to be a new data set.

[3] If volumes were added to a data set for which unit and volume information was retrieved from the catalog, the data set is actually recataloged.

[4] If the step was attempting to receive a passed data set which was new when initially passed, the data set is deleted.

[5] If any job steps reached abnormal termination, the conditional disposition will be processed. Otherwise, the data set is deleted.

[6] If any job steps reached abnormal termination, the conditional disposition will be processed. Otherwise, the data set is kept if it was old when initially passed in the job, or deleted if it was new when originally passed in the job.

List of Exceptions:

- When a nontemporary data set is passed and the receiving step does not assign it a disposition, the system will, upon termination of this step, do one of two things. If the data set was new when it was initially passed, it will be deleted. If the data set was old when initially passed, it will be kept. Temporary data sets are deleted.
- If automatic step restart is to occur, all data sets in the restart step with a status of OLD are kept. All data sets in the restart step with a status of NEW are deleted.
- If automatic checkpoint restart is to occur, all data sets currently in use by the job are kept.
- If a data set is assigned a temporary name or no name, a conditional disposition other than DELETE is invalid. The system assumes DELETE.
- If the data set is not allocated, then no action is taken.

Figure 25. Disposition Processing Chart

Device	Storage Medium	Cylinders	Tracks Per Cylinder	Bytes Per Track	Bytes Per Cylinder	Bytes Per Device (in millions)
2314/2319 (each volume)	Disk	200	20	7,294	145,880	29.17
2305	Disk	Model 1: 48 Model 2: 96	8	Model 1: 14,136 Model 2: 14,660	Model 1: 113,088 Model 2: 117,280	Model 1: 5.4 Model 2: 11.3
3330	Disk	404	19	13,030	247,570	100
3330 Mod 11	Disk	808	19	13,030	247,570	200
3340/3344	Disk	696 (70-megabytes) 348 (35-megabytes)	12	8,368	100,416	69.8 (70-megabytes) 34.9 (35-megabytes)
3350	Disk	555	30	19,069	572,070	317.5
3375	Disk	959	12	35,616	427,392	409.8
3380	Disk	885	15	47,476	712,140	630.2

Note: 3344 pertains only to the 70-megabyte 3340

Figure 26. Direct Access Capacities

DSP	DSP Function	Parameters
CBPRNT	Control Block Print	Yes (note 2)
CI	MVS Converter/Interpreter	Yes (note 2)
DISPDJC	Display Dependent Job Control	Yes (note 2)
DISPLAY	Display Job Queues	No
DJCPROC	Invoke Dependent Job Control Updating	No
DR	Disk Reader	Yes (note 3)
ISDRVR	Input Service Driver (JES3 Control Card Processing)	Yes (Qualified ddname of input file)
JESNEWS	Use JESNEWS Facility	Yes (note 1)
MAIN	Main Service	No
NJPIO	Network Job Processing	Yes (note 1)
OUTSERV	Output Service	No
RI	OS Reader/Interpreter	Yes

Notes:

1. See **OS/VS MVS System Programming Library: JES3**
2. See **OS/VS2 MVS System Programming Library: JES3 Debugging Guide**
3. See **Operator's Library: OS/VS2 MVS JES3 Commands**

Figure 28. Table of Allowable DSPs for PROCESS Statements

REF TAB

The JOB Statement

//Name	Operation	Operand	P/K	Comments
//jobname	JOB	([account number] [,additional accounting information,...])	P	Identifies accounting information. Can be made mandatory.
		[ADDRSPC = {VIRT / REAL}]	K	Requests storage type.
		[CLASS=jobclass]	K	Assigns a job class to each job.
		[COND=((code,operator),...)]	K	Specifies test for a return code.
		[GROUP=group name]	K	Specifies a group associated with a RACF-defined user.
		[MSGCLASS=output class]	K	Assigns an output class for the job.
		[MSGLEVEL=([0 / 1 / 2] [,0 / ,1])]	K	Specifies what job output is to be written.
		[NOTIFY=user identification]	K	Requests a message be sent to a time-sharing terminal.
		[PASSWORD=(password [,new password])]	K	Specifies a password for a RACF-defined user.
		[PERFORM=n]	K	Specifies the performance group a job belongs to.
		[programmer's name]	P	Identifies programmer. Can be made mandatory.
		[PRTY=priority]	K	Specifies a job's priority.
		[RD={R / RNC / NC / NR}]	K	Specifies restart facilities to be used.
		[REGION=valueK]	K	Specifies amount of storage space.
		[RESTART=({* / stepname / stepname.procstepname} [,checkid])]	K	Specifies restart facilities for deferred restart.
		[TIME={([minutes] [,seconds]) / 1440}]	K	Assigns a job a CPU time limit.
		[TYPRUN={HOLD / JCLHOLD / SCAN / COPY}]	K	Holds a job in job queue, scans JCL for syntax errors, or copies the input deck to SYSOUT.
		[USER=userid]	K	Identifies a RACF-defined user.

Legend:

P Positional parameter. (Positional parameters must precede keyword parameters)
K Keyword parameter
{ } Choose one.
[ ] Optional; if more than one line is enclosed, choose one or none.

Figure 29. The JOB Statement

The EXEC Statement

//Name	Operation	Operand	P/K	Comments
// [stepname]	EXEC	[ACCT [.procstepname] = (accounting information,...)]	K	Accounting information for step.
		[ADDRSPC [.procstepname] = {VIRT / REAL}]	K	Requests storage type.
		[COND [.procstepname] = ({(code,operator) / (code,operator,stepname) / (code,operator,stepname.procstepname)} ,... [,EVEN / ,ONLY])]	K	Specifies a test for a return code.
		[DPRTY [.procstepname]=([value1][,value2])]	K	Specifies dispatching priority for a job step.
		[DYNAMNBR [.procstepname] =n]	K	Specifies dynamic allocation.
		[PARM [.procstepname] =value]	K	Passes variable information to a program at execution time.
		[PERFORM [.procstepname] =n]	K	Specifies a performance group for a job.
		[PGM={program name / *.stepname.ddname / *.stepname.procstepname.ddname}]	P	Identifies program.
		[[PROC=] procedure name]	P	Identifies a cataloged or instream procedure.
		[RD [.procstepname] = {R / RNC / NC / NR}]	K	Specifies restart facilities to be used.
		[REGION [.procstepname] =valueK]	K	Specifies amount of storage space.
		[TIME [.procstepname] = {([minutes] [,seconds]) / 1440}]	K	Assigns step CPU time limit.

Legend:

K Keyword parameter.
P Positional parameter. (Positional parameters must precede keyword parameters)
{} Choose one.
[] Optional; if more than one line is enclosed, choose one or none.

Figure 30. The EXEC Statement

REF TAB

The DD Statement

//Name	Oper-ation	Operand	P/K	Comments
//[ddname / procstepname.ddname]	DD	[*]	P	Defines data set in the input stream.
		[AMP=(AMORG / 'BUFND=number' / 'BUFNI=number' / 'BUFSP=number' / 'CROPS=(RCK' / NCK' / NRE' / NRC') / 'OPTCD=(I' / L' / IL') / 'RECFM=(F' / FB' / V' / VB') / 'STRNO=number' / 'SYNAD=modulename' / TRACE)]	K	Completes the access method control block (ACB) for VSAM data sets.
		[BURST=(Y / N)]	K	Specifies whether or not paper output is to go to the Burster-Trimmer-Stacker of the 3800.
		[CHARS=(table name[,table name . . .])]	K	Specifies character arrangement table(s) to be used when printing on the 3800.
		[CHKPT=EOV]	K	For checkpoint at end of volume.
		[COPIES=(nnn[,(group value,group value . . .)])]	K	Requests multiple copies (and grouping, for the 3800 only) of the output data set.
		[DATA]	P	Defines data set in the input stream.
		[DCB=(list of attributes) / DCB=(dsname / *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname [,list of attributes])]	K	Completes the data control block (used for all data sets except VSAM).
		[DDNAME=ddname]	K	Postpones the definition of a data set.
		DEST=(JES2: Nnn / NnnRmmm / Rnnn / RMnnn / RMTnnn / Unnn / LOCAL / name; JES3: ANYLOCAL / device-name / device-address / group-name)	K	Specifies a destination for the output data set.

Figure 31. The DD Statement (Part 1 of 3)

The DD Statement (con't)

//Name	Operation	Operand	P/K	Comments
//[ddname / procstepname.ddname]	DD	DISP=([NEW / OLD / SHR / MOD / ,] [,DELETE / ,KEEP / ,PASS / ,CATLG / ,UNCATLG / ,] [,DELETE / ,KEEP / ,CATLG / ,UNCATLG])	K	Assigns a status, disposition, and conditional disposition to the data set.
		[DLM=delimiter]	K	Assigns delimiter other than /*.
		[DSID=(id[,V])]	K	Indicates to a diskette reader that data is to be merged into the JCL stream at this point or specifies the name to be given to a SYSOUT data set written on a diskette.
		[{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}]	K	Assigns a name to a new data set or to identify an existing data set.
		[DUMMY]	P	Bypasses I/O operations on a data set (BSAM and QSAM).
		[DYNAM]	P	Specifies dynamic allocation.
		[FCB=(image-id [,ALIGN / ,VERIFY])]	K	Specifies forms control information. The FCB parameter is ignored if the data set is not written to a 3211 or 1403 printer.
		[FLASH=(overlay name[,count])]	K	Identifies the forms overlay to be used on the 3800.
		[FREE={END / CLOSE}]	K	Specifies dynamic deallocation.
		[HOLD={YES / NO}]	K	Specifies whether output processing is to be deferred or processed normally.
		[LABEL=([data set seq #] [,SL / ,SUL / ,AL / ,AUL / ,NSL / ,NL / ,BLP / ,LTM / ,] [,PASSWORD / ,NOPWREAD / ,] [,IN / ,OUT] [,EXPDT=yyddd / ,RETPD=nnnn])]	K	Supplies label information.
		[MODIFY=(module name[,trc])]	K	Specifies a copy modification module that is to be loaded into the 3800.

REF TAB

Figure 31. The DD Statement (Part 2 of 3)

The DD Statement (con't)

//Name	Oper-ation	Operand	P/K	Comments
//[ddname / procstepname.ddname]	DD	[MSVGP=(id[,ddname])]	K	Identifies a mass storage group for a mass storage system (MSS) device.
		[OUTLIM=number]	K	Limits the number of logical records you want included in the output data set.
		[PROTECT=YES]	K	Requests RACF protection for tape volumes or for direct access data sets.
		[QNAME=process name]	K	Specifies the name of a TPROCESS macro which defines a destination queue for messages received by means of TCAM.
		[SPACE=({TRK / CYL / blocklength},(primary quantity[,secondary quantity][,directory][,index])[,RLSE][,CONTIG / ,MXIG / ,ALX][,ROUND])]	K	Assigns space on a direct access volume for a new data set.
		[SPACE=(ABSTR,(primary quantity,address[,directory][,index]))]	K	Assigns specific tracks on a direct access volume for a new data set.
		[SUBSYS = (subsystem name[,parm1[,parm2]...[,parm254]])]	K	Specifies the subsystem that will process both the data set and the specified parameters.
		[SYSOUT=(class name[,program name][,form name][,code name])]	K	Assigns an output class to an output data set.
		[TERM=TS]	K	Identifies a time-sharing user.
		[UCS=(character set code[,FOLD][,VERIFY])]	K	Requests a special character set for a 3211 or a 1403 printer.
		[UNIT=(unit address / device type / user-assigned group name[,unit count / ,P][,DEFER]) / UNIT=AFF=ddname]	K	Provides the system with unit information.
		[{VOLUME / VOL}=([PRIVATE][,RETAIN][,volume seq number][,volume count][,SER=(serial number,...) / REF=dsname / REF=*.ddname / REF=*.stepname.ddname / REF=*.stepname.procstepname.ddname])]	K	Provides the system with volume information.

Legend:

P Positional parameter. (Positional parameters must precede keyword parameters)
K Keyword parameter.
{} Choose one.
[] Enclosing subparameter, indicates that subparameter is optional; if more than one line is enclosed, choose one or more.

Figure 31. The DD Statement (Part 3 of 3)

Glossary

The following terms are defined as they are used in this manual. If you do not find the term you are looking for, refer to the Index or to the **IBM Data Processing Glossary**, GC20-1699.

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the American National Standard Vocabulary for Information Processing (Copyright © 1970 by American National Standards Institute, Incorporated), which was prepared by Subcommittee X3K5 on Terminology and Glossary of American National Standards Committee X3. ANSI definitions are marked with an *

address space. The virtual storage assigned to a job, TSO user, or a task initiated by the START command. Each address space consists of the same range of addresses.

affinity. (See volume affinity, unit affinity.)

allocate. To assign a resource for use in performing a specific task.

ASP. Asymmetric multiprocessing system that provides supplimentary support for job management, data management, and task management, performing such functions as scheduling input readers and output writers.

ASP main processor. The MVT or SVM processor that executes jobs assigned to it by a JES3 global processor.

associated data set. Data set on a 3540 diskette volume that is separate from the job stream data set and is to be spooled as a SYSIN data set.

automatic priority group (APG). In VS2, a group of tasks at a single priority level that are dispatched according to a special algorithm that attempts to provide optimum use of CPU and I/O resources by these tasks.

automatic restart. A restart that takes place during the current run, that is, without resubmitting a job; an automatic restart can occur within a step or at the beginning of a step. Contrast with deferred restart.

auxiliary storage. Data storage other than main storage; secondary storage.

backward reference. A facility of the job control language that permits you to copy information from or refer to DD statements that appear earlier in the job.

card image form. Column binary.

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

cataloged data set. A data set that is represented in an index or hierarchy of indexes in the system catalog; the indexes provide the means for locating the data set.

cataloged procedure. A set of job control statements that has been placed in a partitioned data set called the procedure library and that can be retrieved by coding the name of the procedure on an execute (EXEC) statement or started by a START command.

checkpoint data set. A sequential or partitioned data set containing a collection of records (called checkpoint entries) that contain the status of a job and the system at the time the records are written. These records provide the information necessary for restarting a job without having to return to the beginning of the job.

checkpoint restart. The process of resuming a job at a checkpoint within the job step that was abnormally terminated. The restart can be automatic or deferred, where deferred restart involves resubmitting the job. Contrast with step restart.

checkpoint/restart facility. A facility for restarting execution of a program at some point other than at the beginning, after the program was terminated due to a program or system failure. A restart can begin at a checkpoint within a job step or at the beginning of a job step.

command statement. A job control statement, JES2 control statement, and JES3 control statement that is used to issue commands to the system through the input stream.

comment statement. A job control statement used to include information that may be helpful in running a job or reviewing an output listing.

compaction. A method of reducing the number of characters transmitted in a data string to increase communication efficiency. The method represents frequently used characters, (when they occur in pairs), by an eight-bit rather than sixteen-bit binary sequence; each character requires only four bits.

concatenated data sets. A group of logically connected data sets that are treated as one data set for the duration of a job step.

converter/interpreter. The job segment that converts and interprets JCL for the MVS system.

cylinder. The tracks of a disk storage device that can be accessed without repositioning the access mechanism.

data definition (DD) statement. A job control statement that describes a data set associated with a particular job step.

data management. A major function of the operating system that involves organizing, cataloging, locating, storing, retrieving, and maintaining data.

data set. The major unit of data storage and retrieval in the operating system, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

ddname (data definition name). A name assigned to a DD statement. This name corresponds to the ddname appearing in a data control block.

deadline scheduling. An automatic method of controlling a job's scheduling priority to increase the probability of the job being scheduled by a given deadline.

deferred restart. A restart performed by the system on

resubmission of a job by the programmer; deferred restart can begin within a step or at the beginning of a step. Contrast with automatic restart.

delimiter statement. A job control statement used to mark the end of data.

dependent job control (DJC). In JES3, the organizing of a collection of jobs that must execute in a specific order. DJC manages jobs that are dependent upon one another.

device type. A number that corresponds to a type of input/output device. Coding the device type in the UNIT parameter is one way of indicating what input/output device you want allocated to a job step.

directory. A series of 256-byte records at the beginning of a partitioned data set that contains an entry for each member in the data set.

dispatching priority. A number assigned to tasks, used to determine the order in which they will use the central processing unit.

disposition processing. A function performed by the initiator at the end of a job step to keep, delete, catalog, or uncatalog data sets, or pass them to a subsequent job step, depending on the data set status or the disposition specified in the DISP parameter of the DD statement. Disposition processing also occurs at job termination.

dummy data set. A data set for which operations such as disposition processing, input/output operations, and allocation are bypassed.

***dump.** (1) to copy the contents of all or part of storage, usually from an internal storage into an external storage. (2) the data resulting from the process as in (1).

dynamic allocation. Assignment of system resources to a program at the time the program is executed rather than at the time it is loaded into main storage.

dynamic deallocation. Freeing of system resources during program execution rather than at the end of the job.

dynamic support program (DSP). Transient program of JES3 that runs in parallel with the other support functions of JES3.

execute (EXEC) statement. A job control statement that marks the beginning of a job step and identifies the program to be executed or the cataloged or in-stream procedure to be used.

execution priority. A rank assigned to a task that determines its precedence in being selected for execution.

external page storage. The portion of auxiliary storage that is used to contain pages.

external writer. A writer other than JES2 for user-written writer routines and for devices not supported by JES2 or JES3.

folding. A technique used with the universal character set (UCS) feature on an impact printer to allow each of the 256 possible character codes to print some character on a chain or train with fewer graphics. For example, it allows the printing of uppercase graphic characters when lowercase are not available in the character array on the chain or train.

forms control buffer (FCB). A buffer that is used to store vertical formatting information for printing, each position corresponding to a line on the form.

generation data group (GDG). A collection of data sets that are kept in chronological order; each data set is called a generation data set.

generation data set. One generation of a generation data group.

global processor. JES3 processor that controls the job selection for all processors in the system running under JES3.

group name. See user-assigned group name.

HASP. The HASP system provides supplimentary support for job management, data management, and task management, performing functions such as scheduling input readers and output writers.

input service. In JES3 a set of dynamic support programs that read the input data and build the system input data set and control table entries for each job.

input stream. The sequence of control statements and data submitted to the operating system on an input device especially activated for this purpose by the operator.

in-stream procedure. A set of job control statements placed in the input stream that can be used any number of times during a job by naming the procedure on an execute (EXEC) statement.

integrity. Preservation of data or programs for their intended purpose.

JES2 control statement. A statement that controls the input and output processing of jobs run under JES2.

JES3 control statement. A statement that controls the input and output processing of jobs run under JES3.

job. A collection of related problem programs, identified in the input stream by a JOB statement followed by one or more EXEC and DD statements.

job class. Any one of a number of job categories that can be defined by the installation to classify jobs. By classifying jobs and directing initiators to initiate specific classes of jobs, it is possible to control the mixture of jobs that are performed concurrently.

job class queue. A waiting list of job definitions within the input queue in which jobs assigned the same class are arranged in order of priority; jobs with the same class and priority are placed in a first in/first out order.

job control language (JCL). A high-level programming language used to code job control statements.

***job control statement.** A statement in a job that is used in identifying the job or describing its requirements to the operating system.

job journal. Established at JES2 or JES3 initialization to hold restart information for each program in execution.

job library. See private library.

job management. A general term that collectively describes the functions of the job scheduler and master scheduler.

job related output. Output that is neither held nor spun off nor processed by a user-written writer.

job (JOB) statement. The job control statement that identifies the beginning of a job. It contains such information as the name of the job, an account number, and the class and priority assigned to the job.

job step. A unit of work associated with one processing

program or one cataloged procedure and related data. A job consists of one or more job steps.

job step task. A task that is initiated by an initiator according to specifications in an execute (EXEC) statement.

jobname. The name assigned to a JOB statement; it identifies the job to the system.

K. 1024 bytes.

keyword. A symbol that identifies a parameter or subparameter.

keyword parameter. A parameter that consists of a keyword, followed by one or more values.

local devices. Devices attached to the global processor at the central computer center for sending input and receiving output.

local processor. The JES3 (MVS) processor that executes the jobs assigned to it by the global processor.

local station. A station whose control unit is connected directly to a computer I/O channel. Contrast with remote station.

logical record. A record that is defined in terms of the information it contains rather than by its physical traits. You may have to specify the length of the logical record to complete the data control block; one way to specify this is in the LRECL subparameter of the DCB parameter.

logical region. The amount of real storage required by a job a job step to execute efficiently on an ASP main processor when running under JES3.

loosely-coupled multiprocessing. two or more computing systems interconnected by an I/O channel-to-channel adapter. The CPUs can be of different types and have their own unique configurations.

main service. In JES3, a dynamic support program schedules problem programs for execution and manages the flow of data (system input, print, and punch) across the channel-to-channel adapter to and from the global processor.

Mass storage system group (MSVGP). a named collection of mass storage volumes defined by the person in charge of controlling space. Both active and inactive mass storage volumes can be in the group. The volume group is identified by name in JCL on the MSVGP parameter.

mutually exclusive parameters. Parameters that cannot be coded on the same job control statement.

MVS (multiple virtual storage). VS2 Release 2 and all subsequent VS2 releases.

nonpageable dynamic area. An area of virtual storage whose virtual addresses are identical to real addresses; it is used for programs or parts of programs that are not to be paged during execution.

nonsharable volume. A volume that cannot be assigned to two or more data sets.

nonspecific volume request. A request that allows the system to select suitable volumes.

nontemporary data set. A data set that exists after the job that created it terminates.

null statement. A job control statement used to mark the end of a job's control statements and data.

***operating system (OS).** Software which controls the execution of computer programs and which may provide scheduling, debugging, input/output control, accounting, compilation, storage assignment, data management, and release services.

output class. Any one of up to 36 different catagories, defined at an installation, to which output data produced during a job step can be assigned. When an output writer is started, it can be directed to process from one to eight different classes of output data.

***output data.** (SC1) Data to be delivered from a device or program, usually after some processing.

output listing. A form that is printed at the end of a job that can contain such information as job control statements used by the job, diagnostic messages about the job, data sets created by the job, or a dump.

output service. A JES3 service that prints and punches the data sets created during main service.

page. A fixed-length block of instructions, data, or both, that can be transferred between real storage and external page storage.

partitioned data set. A data set in direct access storage that is divided into partitions, called members, each of which can contain a program or part of a program. Each partitioned data set contains a directory (or index) that the control program can use to locate a program in the library.

passed data set. A data set allocated to a job step that is not deallocated at step termination but that remains available to a subsequent step of the same job.

PEND statement. A job control statement used to mark the end of an in-stream procedure.

permanently resident volume. A volume that cannot be physically demounted or that cannot be mounted until it is varied offline (that is, removed from the control of the central processing unit).

physical record. A record that is defined in terms of physical qualities rather than by the information it contains (logical record).

positional parameter. A parameter that must appear in a specified order.

priority. A value assigned to a job that is used to determine when a job is selected for execution.

private library. A user-owned library that is separate and distinct from the system library.

private volume. A mounted volume that the system can allocate only to a data set for which a specific volume request is made.

PROC statement. A job control statement that must mark the beginning of an in-stream procedure; it can also be used, in both cataloged and in-stream procedures, to assign values to symbolic parameters in the procedure.

procedure library. A partitioned data set containing cataloged procedures; the IBM-supplied procedure library is named SYS1.PROCLIB.

procedure step. That unit of work associated with one processing program and related data within a cataloged or in-stream procedure. A cataloged or in-stream procedure consists of one or more procedure steps.

public volume. The term applied to a mounted volume

that the system can allocate to an output data set for which a nonspecific volume request is made. A public volume remains mounted until the device on which it is mounted is required by another volume.

qualified name. A data set name that is composed of multiple names separated by periods (for example, A.B.C.). For a cataloged data set, each name corresponds to an index level in the catalog.

queue. A waiting line or list formed by items in a system waiting for service; for example, tasks to be performed or output to be written by a writer.

reader/interpreter. The job segment that reads and interprets JCL for jobs on ASP main processors.

real storage. The storage of a system/370 computing system from which the central processing unit can directly obtain instructions and data, and to which it can directly return results.

record. A general term for any unit of data that is distinct from all others.

region. In systems with MVS, a subdivision of the dynamic area of main storage set aside for a job step or a system task. You can specify in the REGION parameter on the JOB statement or EXEC statement how large this area of main storage should be.

remote devices. Devices attached to remote work stations for sending input and receiving output.

remote job entry. Submission of job control statements and data from a remote terminal, causing the jobs described to be scheduled and executed as though encountered in the input stream. Also known as remote job processing in JES3.

remote job processing (RJP). The processing of jobs submitted from remote terminals.

remote station. (1) * Data terminal equipment for communicating with a data processing system from a location that is time, space, or electrically distant. (2) Contrast with local station.

reserved volume. A volume that remains mounted until the operator issues an UNLOAD or VARY OFFLINE command.

resource. Any facility of the computer system or operating system required by job or task and includes main storage, input/output devices, the CPU, data sets, and control and processing programs.

restart. The process of resuming a job after it abnormally terminates. When a restart is performed, processing is continued either at the beginning of a job step that caused the abnormal termination or at a checkpoint within this job step.

restart facility. See checkpoint/restart facility.

return code. A value placed in the return code register at the completion of a program. The value is established by the user and may be used to influence the execution of succeeding programs or, in the case of an abnormal end of task, may simply be printed for programmer analysis.

scheduling priority. a rank assigned to a task that determines its precedence in being scheduled.

sequential data set. A data set whose records are organized on the basis of their successive physical positions, such as they are on magnetic tape.

SNA. (Systems Network Architecture) The total description of the logical structure, formats, protocols, and operational sequences for transmitting information units through the communication system.

specific volume request. A request for volumes that informs the system of the volume serial numbers.

spooled data set. A data set written on an auxiliary storage device.

standard job. a JES3 job that consists of input service main service, output service, and purge performed in that order.

step restart. A restart at the beginning of a job step that abnormally terminates. The restart may be automatic (depending on an eligible completion code and the operator's consent) or deferred, where deferred involves resubmitting the job and coding the RESTART parameter on the JOB statement of the resubmitted job.

stepname. The name assigned to an EXEC statement; it identifies a job step within a job.

storage volume. The main function of a storage volume is to contain nontemporary data sets for which a nonspecific volume request was made and PRIVATE was not coded in the VOLUME parameter. A direct access volume becomes a storage volume when so indicated in a MOUNT command or in a member of SYS1.PARMLIB named VATLSTxx.

subsystem. (1) In MVS, any name that is defined via the SUBSYS parameter of the SCHEDULR system generation macro, or any name that is defined in the subsystem names table (IEFJSSNT). (2) A secondary or subordinate system, usually capable of operating independently of, or asynchronously with MVS (for example, the IBM 3800 Printing Subsystem).

symbolic parameter. A symbol preceded by an ampersand that stands for a parameter or the value assigned to a parameter or subparameter in a cataloged or in-stream procedure. Values are assigned to symbolic parameters when the procedure in which they appear is called.

system generation. The process of using an operating system to assemble and link together all of the parts that constitute another operating system.

system library. A partitioned data set named SYS1.LINKLIB that contains programs used by the system.

system messages. Messages issued by the system that pertain to a problem program. These messages appear on an output listing and may include such messages as error messages, disposition messages, and allocation/deallocation messages.

system output device. A device assigned to record output data for a series of jobs.

system resources manager. A group of programs that controls the use of system resources in order to satisfy the installation's performance objectives.

SYS.LINKLIB data set. See system library.

SYS1.PROCLIB data set. See procedure library.

table reference character. A numeric character (0,1,2, or 3) corresponding to the order in which the character arrangement table names have been specified with the CHARS keyword.

task. A unit of work for the central processing unit from

the standpoint of the control program; therefore, the basic multiprogramming unit under the control program.

temporary data set. A data set that is created and deleted in the same job.

temporary library. A library that is created and deleted within a job.

***track.** The portion of a moving storage medium, such as drum, tape, or disk, that is accessible to a given reading head position.

unit. A particular device specified by its unit address, device type, or user-assigned group name.

unit address. The three-character address of a particular device, specified at the time a system is installed; for example, 191 or 293.

unit affinity. A condition under which two or more volumes are located on the same device.

universal character set (UCS) feature. A printer feature that permits the use of a variety of character arrays.

user-assigned group name. Installation defined name to signify a group of devices that may or may not all be of the same type (specified through JCL in the UNIT parameter).

virtual input/output (VIO). Facility to handle temporary data sets that causes them to reside within the paging data sets. To problem program or access method, the data sets appear to reside on some other real direct access storage device.

virtual storage. Addressable space that appears to the user as real storage, from which instructions and data are mapped into real storage locations. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, rather than by the actual number of real storage locations.

volume. That portion of an auxiliary storage device that is accessible to a single read/write mechanism.

volume affinity. A condition under which two or more data sets are located on the same volume.

volume table of contents (VTOC). A table on a direct access volume that describes each data set on the volume.

work station. A terminal device that may or may not be a CPU. At a workstation, an operator can connect into a central system via LOGON, enter jobs, and receive output.

working set. The estimate of bytes of real storage used by the steps of a job.

Index

b use 123
{} braces, use 123
[] brackets, use 123-124
... use 124
++ (instream procedure) 117
++* (instream procedure) 117
+/ (instream procedure) 117
& use 117-118
& & identifying temporary data set 29-30
$A 278
$B 278
$C 278
$D 278
$E 278
$F 278
$G 278
$H 278
$I 278
$L 278
$M 278
$N 278
$P 278
$O 278
$R 278
$S 278
$T 278
$VS 278
$Z 278
* parameter on DD statement 191-192
  data in input stream 191-192
  description 96
  restriction for cataloged or in-stream procedure 110
  retrieving a data set 352
  separating groups of data 191
  (see also DATA and DLM parameters)
* in PGM parameter 166
* in PRIORITY statement 285
* subparameter in RESTART parameter 147-148
*** comment statement 117
*.ddname
  in the DCB parameter 203
  in the DSNAME parameter 226
*.stepname.ddname
  in the DCB parameter 203
  in the DSNAME parameter 226
  in the PGM parameter 166
*.stepname.procstepname.ddname
  in the DCB parameter 203
  in the DSNAME parameter 226
  in the PGM parameter 166
/* delimiter 267
/* for JES2 control statements **(5740-XYS)** 277

ABCMP parameter on NET statement (JES3)
  description 320,321
ABEND dumps
  (see abnormal termination dump)
ABNORMAL parameter on NET statement (JES3)
  description 320-321
  use of 68-69
abnormal termination dump
  JES2 51-52
  JES3 79-80
absolute track technique
  for ISAM data set 102
ABSTR subparameter of SPACE Parameter
  assigning specific tracks 22
  description 248,249
  for ISAM data sets 102
accessing TCAM messages 247
accounting information parameter on JOB statement 130-131
  JES2 parameters for 130
ACCT parameter on EXEC statement 156
ACMAIN parameter on MAIN statement (JES3) 312,316,319
address space 365
address subparameter of SPACE parameter
  assigning specific tracks 22
  description 248-251
address, unit
  on UNIT parameter 259
  specifying unit information 13
ADDRSPC parameter
  on EXEC statement 157
  on JOB statement 132
  requesting storage 9-10
AFF subparameter of UNIT parameter
  description 259,261
  requesting unit affinity 15-16
affinity 365
  unit (see unit affinity)
  volume (see volume affinity)
AL subparameter of LABEL parameter
  (American National standard labels) 238
ALIGN subparameter of FCB parameter 232
alignment of forms 57-58
allocating devices (JES3) 63
  (see also SETUP parameter on MAIN statement)
allocation
  absolute track 102
  definition 365
  devices eligible 11
  dynamic (see DYNAM parameter on DD statement and DYNAMNBR parameter on EXEC statement)
  nonspecific 102
  (see also SPACE parameter on DD statement)
allocation/termination messages
  (see system messages)
alphameric character sets 56,57,84,127
ALX subparameter of SPACE parameter 248,250
American National standard labels 236
AMORG subparameter of AMP parameter 193
AMP parameter on DD statement
  defines VSAM data set 96-99
  description 193-195
ampersand
  as special character 123
  identifying symbolic parameter 117-122,125
  identifying temporary data set 29-30,127
AN character set (1403) 57,84
ANSI printer control characters in RECFM subparameter 214-215
ANSI tape labels 236
APG (automatic priority group) 44,68
apostrophe, specifying 31-32,127
ASCII magnetic tape
  (see DCB parameter)
  (see LABEL parameter)
Assigning values to symbolic parameters
  (see Symbolic Parameters)

associated data sets 31,365
(see also DSID parameter on DD statement)
attributes, DCB
(see DCB subparameters)
AUL subparameter of LABEL parameter
(American National standard and user labels) 238
automatic priority group 44,68,365
(see also priority)
automatic restart
definition 365
use of (JES2) 48
use of (JES3) 76
(see also RD parameter)
automatic step restart (JES2) 48
automatic step restart (JES3) 76
auxiliary storage 365
average block length, space request
(see block length subparameter)
A11 character set (3211) 57,84

backward reference
copying a data set name 31
DDNAME parameter 217
definition 365
DUMMY parameter 229
generation data group 107
VOLUME parameter 262,265
basic direct access method
(see BDAM subparameters of DCB parameter)
basic indexed sequential access method
(see BISAM subparameters of DCB parameter)
basic partitioned access method
(see BPAM subparameters of DCB parameter)
basic sequential access method (BSAM)
(see BSAM subparameters of DCB parameter)
basic telecommunications access method
(see BTAM subparameters of DCB parameter)
BDAM subparameters of DCB parameter 206-216
BFALN subparameter of DCB parameter 206
BFTEK subparameter of DCB parameter 206
BISAM subparameters of DCB parameter 206-216
blank 125
BLKSIZE subparameter of DCB parameter
coded with
* parameter 191
DATA parameter 201
DDNAME parameter 217
description 206
blocklength subparameter of SPACE parameter
description 248,249
requesting space 20
blocks, directory, for BPAM
(see directory)
blocksize (see BLKSIZE subparameter of DCB parameter)
BLP subparameter of LABEL parameter 238,239-240
BPAM subparameters of DCB parameter 206-216
braces {} 123
brackets [] 123-124
buffers
boundary (see BFALN subparameter)
length of (see BUFL subparameter)
number of (see BUFNO subparameter)
for all lines (see BUFSIZE subparameter)
for one line (see BUFMAX subparameter)
for receiving operation (see BUFIN subparameter)
for sending operation (see BUFOUT subparameter)
offset (see BUFOFF subparameter)
type (see BFTEK subparameter)
BSAM subparameters of DCB parameter 206-216
BTAM subparameters of DCB parameter 206-216
BUFIN subparameter of DCB parameter 206
BUFL subparameter of DCB parameter 206
BUFMAX subparameter of DCB parameter 206
BUFND subparameter of AMP parameter 212
BUFNI subparameter of AMP parameter 212
BUFNO subparameter of DCB parameter
coded with
*parameter 191
DATA parameter 201
DDNAME parameter 217
description 206
BUFOFF subparameter of DCB parameter 206
BUFOUT subparameter of DCB parameter 206
BUFSIZE subparameter of DCB parameter 206
BUFSP subparameter of AMP parameter 212
BURST parameter
bursting of output (JES2) 58
bursting of output (JES3) 86
on DD statement 196
on OUTPUT statement (JES2) 282
bypassing data set allocation
defining a dummy data set 93-94
bypassing disposition processing 40
bypassing job initiation (JES2) 47
bypassing label processing 238-240
bypassing I/O operations
(see DUMMY parameter on DD statement)
bypassing job steps
(see COND parameter on EXEC statement)

CALL operator command (JES3) 292
CANCEL operator command
for JCL 266
for JES3 292
capacities of devices
card forms (JES3)
(see FORMS parameter on FORMAT PU statement)
card image form 366
CARDS parameter
on JOBPARM statement (JES2) 279
on MAIN statement (JES3) 312,313,317
CARRIAGE parameter on FORMAT PR statement (JES3)
303,305,307
carriage tape (JES3)
(see CARRIAGE parameter on FORMAT PR statement)
catalog 365
cataloged and in-stream procedures
calling 110,163
changes, allowing for 111
DD statements, modifying 114-115
definition of 110
EXEC statements, modifying 112-114
identifying (in-stream procedure) 110-111
introduction 2
modifying 112-116
passing information (JES2) 45
passing information (JES3) 74
procedure library 111
procedure statement identifying 116-117
restriction on contents of 110
symbolic parameter 117-122
using 112-122
writing 110-111
cataloged data set 365
cataloged procedure
adding to procedure library 111
calling 112,163
DD statements 114-116,176
definition 6,365
description of 110

example of 122
JOBLIB statement 178
selecting a catalog procedure library 46
symbolic parameters 117-122
TIME parameter 172
cataloging a data set
(see CATLG subparameter of DISP parameter)
CATLG subparameter of DISP parameter
description 221,222
use of 36-40
changing a user's password 139
channel program, active
requesting real storage 9
character arrangement tables
used with MODIFY 242
(see also CHARS parameter)
character control
for JES2 56-58
for JES3 83-86
character set codes
for JES2 57
for JES3 84
character sets, description 127
character set, requesting
universal (UCS) 56,84
examples 258
CHARS parameter
on DD statement 197
on FORMAT PR statement (JES3) 303,305,307
on OUTPUT statement (JES2) 282
use of (JES2) 57
use of (JES3) 84
checkpoint data set
(see also SYSCHK and SYSCKEOV DD statements)
checkpoint restart
definition 365
job step checkpoint option (see JOBSTEP parameter on MAIN statement (JES3)
for generation data group 108
use of (JES2) 48
use of (JES3) 76
(see also RD parameter, SYSCHK, and SYSCKEOV DD statements)
checkpoint/restart facility 365
(see also SYSCHK and SYSCKEOV DD statements, RESTART parameter on JOB statement, and RD parameter)
CHKPT macro instruction
RD parameter 144,169-170
use of (JES2) 48
use of (JES3) 76
CHKPT parameter on DD statement
description 198
use of 48,76
CHNGDUMP operator command 266
CHNSIZE subparameter
on FORMAT PR statement (JES3) 303,304,306
on FORMAT PU statement (JES3) 309,309.0,310
CKPTLNS parameter
on OUTPUT statement (JES2) 282
CKPTPGS parameter
on OUTPUT statement (JES2) 282
class (see CLASS parameter)
CLASS parameter
on DATASET statement (JES3) 294
on JOB statement 133
on MAIN statement (JES3) 312,314,317
use of (JES2) 43-44
use of (JES3) 66
class name subparameter of SYSOUT parameter 254
CLOSE subparameter of FREE parameter
description 235
dynamic allocation 25
code name subparameter of SYSOUT parameter 254
code parameter
on OUTPUT statement (JES2) 282
CODE subparameter of DCB parameter 207
comma
continuing control statements 126
leading and trailing commas, caution 120-122
purpose 123,125
when defining symbolic parameters 120-122
command statement
definition 365
for JCL 1,266
for JES2 **(5740-XYS)** 4,278
for JES3 5,292-293
comment statement
definition 365
description 267
introduction 1
comments field
continuing 126
COMPACT subparameter
on FORMAT PR statement (JES3) 303,304,307
on FORMAT PU statement (JES3) 309,310,311
compaction, definition 365
compaction table, specifying 283-284
completion codes 134,158
concatenated data sets 126-127
concatenation
data sets 115-116,126-127
private libraries 92
concurrent use, data set 41
COND parameter
on EXEC statement 158-160
on JOB statement 134
use of (JES2) 47-48
use of (JES3) 73-74
conditional disposition of data set 35-40
(see also DISP parameter on DD statement)
conditional execution of job steps
testing return codes (JES2) 47-48
testing return codes (JES3) 74
(see also COND parameter)
CONTIG subparameter of SPACE parameter
description 248,250
use of 102
contiguous space
requesting 20-23
for indexed sequential data sets 102
continuing control statements 126
control of nontemporary data set
exclusive control 40
shared control 41
CONTROL parameter on FORMAT PR statement (JES3) 303,304,307
control printing (JES2)
length of form and lines per inch 56-58
special character sets 56-57
control program (JES3) 62
(see also TYPE parameter on MAIN statement)
control statements
continuing 126
converter/interpreter 365
COPIES parameter
on FORMAT PR statement (JES3) 303,304.1,307
on FORMAT PU statement (JES3) 309,310
on JOBPARM statement (JES2) 279
on OUTPUT statement (JES2) 282,283,284.1
on SYSOUT DD statement 199
requesting multiple copies (JES2) 55

requesting multiple copies (JES3) 83
copy input deck and bypass job initiation 47
copy modification module
(see MODIFY parameter)
COPY subparameter of TYPRUN parameter 151
(see also bypassing job initiation)
COPYG parameter
on OUTPUT statement (JES2) 282,283
copying the data set name 31
count subparameter of FLASH parameter
on DD statement 234
on OUTPUT statement (JES2) 282,283.0
on FORMAT PR statement (JES3) 303,305,307
CPU time limit
on EXEC statement 172
on JOB statement 149
CROPS subparameter of AMP parameter 193-194
CPRI subparameter of DCB parameter 207
CYL (cylinders) subparameter of SPACE parameter
description 248,249
space request 20-23
cylinders
definition 359
number per device 358
(see also CYL subparameter of SPACE)
CYLOFL subparameter of DCB parameter 207

data
identifying 9-42
modifying 56,83
data control block
(see DCB parameter)
data definition (DD) statement 365
(see also DD statement)
data management 365
DATA parameter on DD statement
creating a data set 350-351
data in input stream 96,110
description 201
restrictions 201
retrieving a data set 352
separating groups of data 201
(see also * and DLM parameter)
data set, non-VSAM
cataloging 38
creating 350-351
definition 365
delaying writing of (JES2) 54
delaying writing of (JES3) 81-82
deleting 37
disposition processing 35-40
exclusive control of 40
extending 353
identifying to system 27-35
integrity 41
keeping 37-38
multivolume 13
nontemporary, creating 28-29
nontemporary, retrieving 28-29
passing 38-39
parameters 27
retrieving 352
shared control of 41
temporary, creating 29-30
temporary, retrieving 29-30
uncataloging 38
writing output (JES2) 52-58
writing output (JES3) 80-86
data set disposition
(see DISP parameter on DD statement)
data set integrity 41
data set label
completing the data control block 203-204
copying attributes from 204
model 105
data set name
copying 31
in apostrophes 31-32
special characters, rules of 31-32,227
temporary 227
nontemporary 227
data set organization
(see DSORG subparameter of DCB parameter)
data set sequence number subparameter on LABEL parameter 32,238,239-240
data set status
description 221
use of 36
DATASET statement for JES3
description 294-295
introduction 5
DCB macro instruction
coding the DCB parameter 203
requesting exclusive control of part of a data set 40
DCB parameter on DD statement
for dummy data set 93
definition 203-205
description by access method 203-205
indexed sequential data sets 100-101,104
modifying in cataloged or instream procedures 115-116
for private libraries 91
BDAM subparameters 206-216
BISAM subparameters 206-216
BPAM subparameters 206-216
BSAM subparameters 206-216
BTAM subparameters 206-216
EXCP subparameter 206-216
GAM subparameters 206-216
QISAM subparameter 206-216
QSAM subparameters 206-216
TCAM subparameters 206-216
DD statement
description 175-176
data in input stream, restriction 110
definition 362-364,365
introduction 1-2
maximum per job 175
special ddnames 175
ddname 365
DDNAME parameter
on DATASET statement (JES3) 294
on DD statement 27,217
on FORMAT PR statement (JES3) 303,303.0,306
on FORMAT PU statement (JES3) 309,310
DEADLINE parameter on MAIN statement (JES3) 312,315,318
deadline scheduling (JES3)
definition 365
use of 67
(see also DEADLINE parameter on MAIN statement)
deallocation, dynamic
(see FREE parameter on DD statement)
dedicated devices
dependent job control 68-69
(see also DEVPOOL parameter on NET statement (JES3))
default disposition processing 40
DEFER subparameter of UNIT parameter
description 259,260
use of 14

deferred mounting
(see DEFER subparameter of UNIT parameter)
deferred restart
definition 366
use of (JES2) 48
use of (JES3) 76
(see also RD parameter on JOB and EXEC statements)
DEL statement (JES2) 277
DELAY operator command (JES3) 292
delaying job initiation (JES2) 46-47
TYPRUN=HOLD parameter 151
SETUP control statement (JES2) 288
delaying the writing of an output data set (JES2) 54
delaying the writing of an output data set (JES3) 81-82
DELETE subparameter of DISP parameter
description 221,222
use of 37
deleting a data set
(see DELETE subparameter of DISP parameter)
deleting records
exclusive control 40
deleting unused space
(see RLSE subparameter of SPACE parameter)
delimiter statement
definition 367
description 269
input stream data 96
introduction 1
(see also DLM, DATA, and * parameters)
delimiter other than /*
(see DLM parameter on DD statement)
demounting a tape volume
(see VOLUME parameter on DD statement)
DEN subparameter of DCB parameter 207
dependent job control
dedicating devices to 68-69
definition 366
dependencies between jobs in different nets 68-69
examples of 70-72
how to code 70
when used 68-69
(see also NET statement (JES3))
DEST parameter
on DD statement 236
on FORMAT PR statement (JES3) 303,304,306
on FORMAT PU statement (JES3) 309,310
on OUTPUT statement (JES2) 282,283-284
on XMIT statement (JES2) 290.2
output destination (JES2) 58-59
output destination (JES3) 86-87
(see also ROUTE statement (JES2))
destination (JES2)
(see DEST parameter on DD and OUTPUT statements and ROUTE statement)
destination (JES3)
(see DEST parameter on DD and FORMAT statements)
(see also ROUTE statement)
destination (TSO)
(see DEST parameter on DD statement)
destination (VM)
(see DEST parameter on DD statement)
devices
capacities 358
for dependent job control 69
JES3 setup 63-64
specifying JES2 53
valid for VSAM 97
device type
definition 366
list of found in *SPL: System Generation Reference* 259
subparameter of UNIT parameter 259-261
use of 13-14
DEVPOOL parameter on NET statement (JES3)
description 320,321,322
use of 69
DEVRELSE parameter on NET statement (JES3)
description 320,321,323
use of 69
DIAGNS subparameter of DCB parameter 207
dispatching priority
assigning 44,68
APG default 44,68
execution priority (JES3) 68
direct access capacities 358
direct access data sets
disposition processing 36-37,39-40
shared control 41
direct access device 10
direct access volumes
for partitioned data sets 22
for passed data sets 38-39
directory
definition 22,366
subparameter of SPACE parameter 248,249,250
DISABLE operator command (JES3) 292
DISP parameter on DD statement
description 221-223
disposition processing, non-VSAM 35-42
for generation data set 106,108
for private library 90-91
for ISAM data set 102,104
on JOBLIB DD statement 178-180
on STEPLIB DD statement 182-184
on SYSABEND DD statement 185
on SYSCHK DD statement 188-189
on SYSCKEOV DD statement 190
on SYSMDUMP DD statement 185
on SYSUDUMP DD statement 185
dispatching priority
assigning (JES2) 44
assigning (JES3) 68
definition 366
use of APG (JES2) 44
use of APG (JES3) 68
(see also DPRTY parameter on EXEC statement)
DISPLAY operator command 266
disposition, non-VSAM data set
bypassing 40
conditional 35-37
default 40
processing of 35-42,357,366
specifying 35,36
unreceived passed 39-40
DLM parameter on DD statement
description 224
input stream data 96
on XMIT statement (JES2) **(5740-XYS)** 290.2
(see also DATA and * parameters and delimiter statement)
DPRTY parameter on EXEC statement
calculating dispatching priority (JES2) 44
calculating dispatching priority (JES3) 68
description 161
DSID parameter on DD statement
associated data sets 31
creating a data set 350
description 225
retrieving a data set 352
DSN parameter on DD statement
(see DSNAME parameter)
DSNAME parameter on DD statement
adding members to a private library 90
apostrophes, specifying 31-32
description 226-228
for generation data set 29,106,107

for private library 90-91
for ISAM data sets 100,103
members of a partitioned data set 29
nontemporary data set 28,227
on JOBLIB DD statement 178,179
on STEPLIB DD statement 182,183
on SYSABEND DD statement 185
on SYSCHK DD statement 188
on SYSCKEOV DD statement 190
on SYSMDUMP DD statement 185
on SYSUDUMP DD statement 185
special characters, specify 227
specifying DSNAME 28
storing a dump (JES2) 51-52
storing a dump (JES3) 80
temporary data set 29-30,227
DSNAME=NULLFILE 93
DSORG subparameter of DCB parameter
description 208
requesting space for an index 22
DSP (JES3)
allowable on PROCESS statement 359
definition 366
dsp parameter on PROCESS statement (JES3)
dummy data set
bypassing disposition processing 40
bypassing I/O operations (JES2) 54
bypassing I/O operations (JES3) 81-82
defining 93-94
definition 366
reading 94
suppressing output (JES2) 54
suppressing output (JES3) 82
writing 94
DUMMY parameter on DD statement
backward references 229
creating a dummy data set 93
description 229-230
nullifying a procedure 229
suppressing output (JES2) 54
suppressing output (JES3) 82
dump, abnormal termination
definition 366
high-density 197
requesting (JES2) 51-52
requesting (JES3) 79-80
storing 185
DUMP value on CHARS parameter 197
DYNAM parameter on DD statement
description 231
dynamic allocation 25
dynamic allocation 366
(see also DYNAM parameter on DD statement and DYNAMNBR parameter on EXEC statement)
dynamic deallocation 366
(see also FREE parameter on DD statement)
dynamic support program (DSP) 366
DYNAMNBR parameter on EXEC statement
description 162
dynamic allocation 25

ellipsis 124
ENABLE operator command (JES3) 292
END subparameter of FREE parameter
description 235
dynamic deallocation 25
enqueuing on a data set 40
ENDDATASET statement (JES3)
description 296
introduction 5
ENDPROCESS statement (JES3) 297
EOF statement (JES2) 277
ERASE operator command (JES3) 296
EROPT subparameter of DCB parameter 208
error option
(see EROPT subparameter of DCB parameter)
esoteric name
(see user-assigned group name)
EVEN subparameter of COND parameter
description 158-159
use of (JES2) 47
use of (JES3) 72
Examples
dependent job control (JES3) 70-72
disposition processing 42
dynamically deallocating data sets 26
generation data sets 109
identifying data sets 35
networking (JES3) **(5740-XYN)** 77
obtaining output (JES2) 51-60
obtaining output (JES3) 77
requesting space 23
requesting storage 10
requesting units and volumes 19
routing a job (JES2) 49
routing a job (JES3) 77
spool partitioning (JES3) **(5740-XYN)** 63,63.1
unit and volume affinities 18
exclude particular processors (JES3) 62
EXCP (execute channel program) subparameters of DCB parameter 206-216
EXEC statement
cataloged procedure, use with 110
description 155,361,366
introduction 1-2
modifying parameters on 112-114
parameters, keyword 155
parameters, positional 155
restriction on changing PGM parameter 113
execute statement (EXEC)
(see EXEC statement)
execute channel program
(see EXCP subparameters of DCB parameter)
existing data sets
default disposition processing 40
volume request 11
EXPDT subparameter of LABEL parameter
description 238,239,240
use of 34-35
EXPDTCHK parameter on MAIN statement (JES3) 312,314,317
expiration date
when DELETE is coded 36-37
when KEEP is coded 37-38
(see also LABEL parameter on DD statement)
expiration date checking (JES3) (see EXPDTCHK parameter on MAIN statement)
explicit setup (JES3) (see SETUP parameter on MAIN statement)
extending a data set
additional space 21
multiple units 14
external page storage 366
(see also virtual storage)
external writer 366
(see also PR parameter on JES3 FORMAT statement)
EXTWTR subparameter (JES3) **(5740-XYN)**
of PR parameter on JES3 FORMAT statement 303,304,306
of PU parameter on JES3 FORMAT statement 309,309.0,311

FAIL operator command (JES3) 292
FAILURE parameter on MAIN statement (JES3) 312,314,317
FCB parameter
  definition 366
  on DD statement 232-233
  on FORMAT PR statement (JES3) 303,305,307
  on OUTPUT statement (JES2) 282,283.0,284.1
  use of (JES2) 57-58
  use of (JES3) 83,85
FETCH parameter on MAIN statement (JES3) 312,315,318
fetching devices
  (see FETCH parameter on MAIN statement (JES3))
fixed-length record
  (see RECFM subparameter of DCB parameter)
FLASH parameter
  on DD statement 234
  on FORMAT PR statement (JES3) 303,305,307
  on OUTPUT statement (JES2) 282,283.0,284.1
  use of (JES2) 58
  use of (JES3) 86
FLASHC parameter
  on OUTPUT statement (JES2) 282,283
FOLD subparameter of UCS parameter 257
folding 366
  (see also FOLD subparameter of UCS parameter)
form name subparameter of SYSOUT parameter
  description 254
  use of (JES2) 56
  use of (JES3) 83
FORMAT statement (JES3)
  description 298
  introduction 5
  PR (print) 83-85,298,303-308
  PU (punch) 86,298,309-310
  use of 83-87
FORMAT PR statement (JES3)
  description 298,303-308
  output 82-85
FORMAT PU statement (JES3)
  description 298,309-310
  output 86
forms (see FORMS parameter and form name subparameter)
forms control buffer feature (see FCB parameter)
form and character control
  requesting (JES2) 56-58
  requesting (JES3) 83-86
forms control (see FCB and FORMS parameters)
forms overflow (JES3) (see OVFL parameter on FORMAT PR statement)
forms overlay
  (see FLASH parameter)
FORMS parameter
  on FORMAT PR statement (JES3) 303,305
  on FORMAT PU statement (JES3) 309
  on JOBPARM statement (JES2) 279
  on OUTPUT statement (JES2) 282,283.0
  use of (JES2) 56
  use of (JES3) 83
FREE operator command (JES3) 292
FREE parameter on DD statement
  description 235
  dynamic deallocation 25
FRID subparameter of DCB parameter 208
FROM parameter on FORMAT NJP statement (JES3) 302
FUNC subparameter of DCB parameter 208
GAM subparameters of DCB parameter 206-216
GDG (generation data groups)
  building a base entry 105
  creating 105
  creating a model data set label 105-106
  data set definition 365
  definition 366
  examples of creating and retrieving 109
  generations of 29
  parameters, creating a GDG 106-107
  parameters, retrieving a GDG 107-108
  restarting a job 108
  retrieving 107-108
generation data groups
  (see GDG)
generation data set 366
  (see also GDG)
generation numbers, relative 29,105
GETMAIN (see ADDRSPC and REGION parameters)
generic name
  (see device type)
global processor
  definition 366
  selecting 62
  (see also SYSTEM parameter on MAIN statement (JES3))
GNCP subparameter of DCB parameter 208
graphics access method
  (see GAM subparameters of DCB parameter)
graphic device
  eligible for allocation 10
group name 366
  (see also user-assigned group name)
group name subparameter
  of GROUP parameter on JOB statement 135
  of ORG parameter on JES3 MAIN statement 312,315
GROUP parameter on JOB statement 135
group value subparameter of COPIES parameter
  on DD statement 199
  on FORMAT PR statement (JES3) 303,304,307
  on OUTPUT statement (JES2) **(5740-XYS)** 282,283
  use of 55,83-85
G11 character set (3211) 57,84

HASP 366
high-density dump
  requesting (JES2) 52
  requesting (JES3) 80
  used with CHARS parameter 197
  used with FCB parameter 232
high watermark setup 63-64
  (see also SETUP parameter on MAIN statement (JES3))
HN character set (1403) 57,84
HOLD operator command 266
HOLD parameter
  delaying the writing of an output data set (JES2) 54
  delaying the writing of an output data set (JES3) 81-82
  on DD statement 237
  on MAIN statement (JES3) 312,313
HOLD subparameter of TYPRUN parameter 151
hold output (see HOLD parameter)
H11 character set (3211) 57,84

IBM standard labels 32-34
  (see also LABEL parameter)
IEALIMIT program 132,146,157,171
IEBIMAGE program
  for character arrangement tables 58,84
  for copy modification modules 56,83,242

IEFBR14 program 46,75
image for printing a data set, requesting 57,84
image identifier
(see FCB parameter)
IN subparameter of LABEL parameter
description 238-240
use of 34
incremental quantity
(see secondary quantity)
index description on OUTPUT statement (JES2) 283,284
description on SPACE parameter 248,249
used with indexed sequential data set 22
INDEX parameter
on OUTPUT statement (JES2) 282,283,284,284.2
index print position 282,283
index area, specifying space for 22
indexed sequential data set
area arrangement 103,355
creating 22,100-103,354
creating, example of 104-105
general 100
index space 22
parameters, creating 100-103,354
parameters, retrieving 103-104,354
retrieving 103-104,354
retrieving, example of 104-105
specific tracks 22
temporary 30
INOUT specification (OPEN macro instruction)
for BSAM, overriding 34
input stream
definition 6-8,366
entering data 6,96,191
entering commands 266
input/output operations, bypassing 53-54,81-82
input service 366
input stream 366
INQUIRY operator command (JES3) 292
installation-written writer routine 53-54
in-stream procedure
definition of 6,366
description of 110-122
example of 111,122
passing information 6
symbolic parameters 117-122
(see also PEND and PROC statements)
INT parameter on FORMAT PU statement (JES3)
309,310
integrity, data set
definition 366
how system processes 41
insuring 40
Interactive Problem Control System
requesting an abnormal termination dump (JES2) 52
requesting an abnormal termination dump (JES3) 80
on SYSMDUMP DD statement 185
interpret card output (JES3) 86
INTRDR on SYSOUT DD statement 254
introduction 1-8
INTVL subparameter of DCB parameter 208
IORATE parameter on MAIN statement (JES3)
description 312,315
use of 66
IPCS (see Interactive Problem Control System)
ISAM data set
(see indexed sequential data set)

J parameter on DATASET statement (JES3) 294,295
JCL statements
fields of 124-125
how to code 123-128
introduction 1-2
no longer supported iii
requesting listings of (JES2) 51
requesting listings of (JES3) 79
JCLHOLD subparameter of TYPRUN parameter 151
JCLTEST 75
JESJCL 299
JESMSG 299
JES2 operator commands 278
JES2 statements
coding 277
description 277-290,366
example of 7-8
introduction 4
scheduling a job 43-45
JES3 statements
definition 366
description 291-327
how to code 291
introduction 5
setup 63.1-65
job
assigning class and priority (JES2) 44
assigning class and priority (JES3) 66
definition 366
delaying initiation (JES2) 46-47
delaying initiation (JES3) 67
in input stream 6
introduction 1-8
job class 366
(see also CLASS parameter)
job class queue 366
job control language
definition 366
introduction 1-2
job control language statements 366
(see also JCL statements)
job entry subsystem (2)
(see JES2 statements)
job entry subsystem (3)
(see JES3 statements)
job failure
JES2 recovery 48
JES3 recovery 76
job initiation, delaying (JES2) 46
job initiation, delaying (JES3) 67
job journal 366
(see also JOURNAL parameter on MAIN statement (JES3))
job library 366
(see also JOBLIB DD statement)
job log (JES2) 279
job management 366
job performance (see PERFORM parameter)
job priority (see priority)
job related output 366
job scheduling
deadline scheduling (JES3) 67
improving 61-62
using JES2 and JCL statements 43-45
using JES3 and JCL statements 61-62
job selection (JES2) 43-45
job selection (JES3) 66-69
(see also MAIN statement (JES3))
job setup
(see SETUP parameter on MAIN statement (JES3))
JOB statement
description 129,360,366
examples of 129
introduction 1,2

parameters, keyword 129
parameters, positional 129
job step
definition 366
dispatching priority (JES2) 44
dispatching priority (JES3) 68
in input stream 6-8
introduction 1
performance 44-45,67
job step task 367
JOBCAT DD statement
description 177
master catalog 92
private catalog 92
VSAM 96
jobclass subparameter of CLASS parameter 133
JOBLIB DD statement
concatenating private libraries 92
creating a private library 81-90
description 178
effect with STEPLIB DD statement 91
placement of statement in job 89-90
retrieving a private library 91
when COND=ONLY is coded 159
jobname 367
JOBPARM statement (JES2)
COPIES parameter 55-56
description 279-280.1
FORMS subparameter of OUTPUT statement 282,283
introduction 4
JOURNAL parameter on MAIN statement (JES3) 312,313,314,317

K 367
KEEP subparameter of DISP parameter
description 221,222
use of 37-38
keeping a data set
(see KEEP subparameter of DISP parameter)
KEYLEN subparameter of DCB parameter 209
keylength
(see KEYLEN subparameter of DCB parameter)
keyword 367
keyword parameters
definition 125,367
on DD statement 176
on EXEC statement 155
on JOB statement 129

label (see LABEL parameter on DD statement)
LABEL parameter on DD statement
description 238-241
for indexed sequential data sets 101
for generation data sets 107,108
retention period 37-38
use of 32-35
label type on LABEL parameter
description 238,239-240
use of 32-35
length restriction
for symbolic parameters 119
lengthening a data set
additional space 20-22
exclusive control 40
multivolume data set 13
(see also MOD subparameter of DISP parameter)
library
placing a cataloged procedure in 111
private 89-92
temporary 89,92
(see also JOBLIB and STEPLIB DD statements)
LIMCT subparameter of DCB parameter 209
limiting output records
(see OUTLIM parameter on DD statement)
LINDEX parameter
on OUTPUT statement (JES2) 282,283,284,284.2
LINECT parameter
on JOBPARM statement (JES2) 279
on OUTPUT statement (JES2) **(5740-XYS)** 282,284,284.2
LINES parameter
on JOBPARM statement (JES2) 279
on MAIN statement (JES3) 312,313,317
listings of JCL statements and system messages
requesting (JES2) 51
requesting (JES3) 79
(see also MSGCLASS and MSGLEVEL parameters on JOB statement)
local devices 367
local processor 367
(see also SYSTEM parameter on MAIN statement (JES3))
local station 367
LOCAL subparameter
on OUTPUT statement (JES2) 282
on ROUTE statement (JES2) 286
LOG operator command 266
logical record length 367
(see also LRECL subparameter of DCB parameter)
logical region 367
loosely-coupled multiprocessing
definition 367
description 61
LRECL subparameter of DCB parameter 209
LREGION parameter on MAIN statement (JES3) 312,315,318
LTM subparameter of LABEL parameter 238

main service 367
MAIN statement (JES3)
deadline scheduling 67
description 312-319
introduction 5
job processing balance 67
job setup 63-65
output 86-87
updating a procedure library 111
mass storage system
definition 367
mass storage volume groups 24-25
space requests 25
volume requests 24-25
MSVGP requests 243
mass storage volumes
specifying SPACE parameter 25
specifying VOLUME parameter 24
specifying MSVGP parameter 243
mass storage volume groups 24-25
(see also MSVGP parameter)
memory
(see address space)
message class parameter
(see MSGCLASS parameter on JOB statement)
message level parameter
(see MSGLEVEL parameter on JOB statement)
MESSAGE operator command (JES3) 292
message queue records
(see THRESH subparameter of DCB parameter)
MESSAGE statement (JES2)

description 281
introduction 4
message, system 51,79
minimum region size 10
MOD subparameter of DISP parameter
description 221,223
use of 36
MODE parameter on DATASET statement (JES3) 294,295
MODE subparameter of DCB parameter 209
mode for card reader/punch
(see MODE subparameter of DCB parameter)
MODIFY operator command
for JCL 266
for JES3 292
MODIFY parameter
on DD statement 242
on FORMAT PR statement (JES3) 303,306,307
on OUTPUT statement (JES2) 282,283,284.2
use of (JES2) 56
use of (JES3) 83
modifying a cataloged procedure 112-116
(see also UPDATE parameter on MAIN statement (JES3))
MODTRC parameter
on OUTPUT statement (JES2) 282,283
module name subparameter of MODIFY parameter
on DD statement 242
on FORMAT PR statement (JES3) 303,306
on OUTPUT statement (JES2) 282,283
MONITOR operator command 266
MOUNT operator command 266
MSGCLASS parameter on JOB statement
description 136
printing output (JES2) 51
printing output (JES3) 79-86
MSGLEVEL parameter on JOB statement
description 137
listing JCL and messages (JES2) 51
listing JCL and messages (JES3) 79
MSS (see mass storage system)
MSS parameter on MAIN statement 312,316
MSVGP parameter on DD statement
defining mass storage volumes 24-25
description 243
multiple copies, output data set
(see COPIES parameter)
multiple units 14
multivolume data sets 13
mutually exclusive parameters
definition 367
table of 356
used to override a parameter in a procedure 114
MXIG subparameter of SPACE parameter 248,250

N parameter (JES2) **(5740-XYS)**
on ROUTE statement xx
on XEQ statement xx
N subparameter of BURST parameter
on DD statement 196
on OUTPUT statement (JES2) 282
N subparameter of DEST parameter (JES2) 219
name of a data set 28
name field in control statements 124
national character set 127
NC subparameter of RD parameter
on EXEC statement 169,170
on JOB statement 144
NCK subparameter of AMP parameter 193
NCP subparameter of DCB parameter 209
NET statement for JES3
description 320-323
introduction 5
dependent job control 68-72
how to code 70
examples of 70-72
NETACCT statement (JES2)
description 281.0
example 281.0
introduction 4
NETACCT statement (JES3) **(5740-XYN)**
description 323.0
example 323.0
introduction 5
NETID parameter on NET statement (JES3)
description 320,322
use of 68-72
NETREL parameter on NET statement (JES3)
description 320,322
use of 68-70
Networking (JES3) **(5740-XYN)**
introduction 87
description 62-62.1
(see also JES3 //*ROUTE XEQ)
NEW subparameter of DISP parameter
description 221
exclusive control 40
use of 36-40
new data sets
default disposition 40
on direct access devices 20
specifying status of 36
volume request 11-12
new password subparameter of PASSWORD parameter 139
NHOLD parameter on NET statement (JES3)
description 320,321
use of 68-72
NL subparameter of LABEL parameter 238,239
NO subparameter of HOLD parameter 237
node, subparameter of DEST parameter 219,220
node, transmitting data to
(see JES2 /*XMIT statement)
nodename.userid subparameter
on NOTIFY statement (JES2) 281.1
on OUTPUT statement (JES2) 282
on ROUTE statement (JES2) 286
on JES3 FORMAT statement
of PR parameter 303
of PU parameter 309
on JES3 MAIN statement
of ORG parameter 312,315
on JES3 ROUTE XEQ statement 328
NOLOG parameter
on JOBPARM statement (JES2) 279
nonpageable dynamic area 10,52,368
nonsharable volume
definition 367
for multivolume data set 13
nonspecific volume requests
definition 367
mass storage volumes 24-25
space requests for 11
types of 12
nonstandard job processing (JES3) (see PROCESS statement)
nonstandard labels 32-34
(see also LABEL parameter on DD statement)
nontemporary data set 27-29,367
NOPWREAD subparameter of LABEL parameter
description 238,240
use of 34
normal disposition of data sets 36-40

NORMAL parameter on NET statement (JES3)
description 320-321
use of 68-72
notational conventions 123,124
NOTIFY parameter on JOB statement 138
NOTIFY statement (JES2)
description 281.1
example 281.1
introduction 4
NR subparameter of RD parameter
on EXEC statement 169
on JOB statement 144
NRC subparameter of AMP parameter 193,194
NRCMP parameter
description 321
on NET statement (JES3) 320
NRE subparameter of AMP parameter 193,194
NSL subparameter of LABEL parameter 238
NTM subparameter of DCB parameter 209
null statement
definition 367
description 270
introduction 1
NULLFILE, assign to DSNAME 93
(see also DUMMY parameter)
nullifying parameters in a procedure
DCB parameter 115
DUMMY parameter 115
on DD statements 114-116
on EXEC statements 112-114
symbolic parameters 120

old data sets
(see OLD subparameter of DISP parameter)
OLD subparameter of DISP parameter
description 221
use of 36-40
ONLY subparameter of COND parameter
on EXEC statement 158-159
use of 48,73
OPEN/CLOSE/EOV trace option
(see DIAGNS subparameter of DCB parameter)
operand field in control statements 125
operating system (OS) 367
operation field in control statements 124-125
operator commands
for JCL 266
for JES2 278
for JES3 292
OPERATOR statement for JES3
description 324
introduction 5
operator subparameter of COND parameter
on EXEC statement 158
on JOB statement 134
OPHOLD parameter on NET statement (JES3)
description 320,321
use of 68-72
OPTCD subparameter
of AMP parameter 192,194
of DCB parameter 210-211
optional services
(see OPTCD subparameter)
ORG parameter on MAIN statement (JES3) 312,315,318
OUT subparameter of LABEL parameter
description 238,240
use of 34
OUTIN specification for BSAM, overriding 34
OUTLIM parameter on DD statement
description 245
limiting output records (JES2) 54
limiting output records (JES3) 82
output (JES2)
assigning to classes 52-53
cataloged and in-stream procedures 116-117
data sets 53-58
delaying 54
for 3800 printer (see OUTPUT statement)
limiting 54
listing JCL messages 51
obtaining 51-60
printing dumps 51-52
writing 52-58
output (JES3)
assigning to classes 80-81
cataloged and in-stream procedures 116-117
data sets 80-86
delaying 81-82
for 3800 printer (see FORMAT PR statement)
in a network 62-62.1
limiting 82
listing JCL messages 79
obtaining 80-86
printing dumps 79-80
remote job processing 87
writing 80-86
output classes (JES2)
assigning data sets to 52-53
assigning messages to 51
definition 367
output classes (JES3)
assigning data sets to 80-81
assigning messages to 79
definition 367
output class subparameter of MSGCLASS parameter
description 136
use of (JES2) 51,52-53
use of (JES3) 79,80-81
output data 367
output data set
allocating space for (see SPACE parameter)
conditional disposition (see COND and DISP parameters)
copies of (see COPIES parameter)
creating 350-351
disposition (see DISP parameter)
lengthening 248-251,353
routing of (see SYSOUT parameter)
stackers for
(see BURST and STACKER parameters)
status (see DISP parameter)
unit information (see UNIT parameter)
volume information (see VOLUME parameter)
with UCS feature (see UCS parameter)
output device
(see system output device)
output form (see FORMS parameter and form name subparameter)
output listing
definition 367
dumps 51-52,79-80
identifying cataloged and in-stream procedures 116-117
JCL statements 51,79
suppressing of (JES2) 54
suppressing of (JES3) 82
system messages (JES2) 51
system messages (JES3) 79
output service 367
OUTPUT statement (JES2)
description 282-284
introduction 4

use of 56-57
override
of catalog procedures 6,112-116
of original secondary quantity request for space 21
of symbolic parameters 118-120
parameters in a procedure
on a DD statement 114-116
on an EXEC statement 112-114
overflow area
for indexed sequential data set 100
overlay name subparameter of FLASH parameter
on DD statement 234
on FORMAT PR statement (JES3) 303,305,307
on OUTPUT statement (JES2) 282,283
OVFL parameter on FORMAT PR statement (JES3)
description 303,305,307
use of 85

P subparameter of UNIT parameter
description 259
use of 14
page 10,367
PAGEADD operator command 266
parameters
adding, nullifying, overriding 112-122
in cataloged or in-stream procedure 112-116
notation for defining 125
symbolic 117-122
on DD statement 114-116
on EXEC statement 112-114
parentheses
to enclose subparameter list 125
inclusion in variables 119-122
PARM parameter on EXEC statement
description 163-164
modified in procedure 112-114
passing information 45
partition name subparameter
(see SPART parameter)
partitioned data set
absolute track allocation 22
adding member to 29
concatenating 115-116
creating 28-29,350
definition 367
directory space for 22
lengthening 353
member name 29
nontemporary name 28-29
retrieving a member of 28-29,352
temporary name 29-50
PASS subparameter of DISP parameter
description 221-240
use of 36,38-39
passed data set 367
(see also PASS subparameter of DISP parameter)
passing information to a job (JES2) 45-46
passing information to a job (JES3) 74
passing a private library 89-91
(see also JOBLIB DD and STEPLIB DD statements)
PASSWORD parameter on JOB statement 139
password protection
(see PASSWORD and NOPWREAD subparameters of LABEL parameter)
PASSWORD subparameter of LABEL parameter
description 238,240
use of 34
password subparameter of SIGNON statement 290,346
PAUSE statement (JES3)
description 325
introduction 5
PCAN character set (1403) 57,84
PCHN character set (1403) 57,84
PCI subparameter of DCB parameter 211
PEND statement
definition 367
description 273
introduction 1
use of 110-111
performance (see PERFORM parameter)
PERFORM parameter
on EXEC statement 165
on JOB statement 140
use of (JES2) 44-45
use of (JES3) 67
permanently resident volume
definition 367
private volume not demounted 12
PGM parameter on EXEC statement
description 166-167
for private library 89
identifying the program 45-46,74
JCLTEST 75
restriction, cataloged procedures 113
physical record 367
PN character set (1403) 57,84
positional parameters
definition 125,367
DUMMY parameters 93
on DD statement 175
on EXEC statement 155
on JOB statement 129
in operand field 125
positioning, multivolume 13
PR parameter on FORMAT statement (JES3) 298,303-308
predecessor job, dependent job control 68-72
primary quantity subparameter of SPACE parameter
description 248,249
satisfying request 20-21
prime area 100,226
PRINT parameter
on FORMAT statement (JES3) 299,300
on ROUTE statement (JES2) 286
print output (JES3) (see FORMAT PR statement)
printer
on OUTPUT statement (JES2) 282
on ROUTE statement (JES2) 285
printer form control
for JES2 56
for JES3 83
printer train (JES3) 84
printers
character sets for 57,84
images, requesting 57,84
(see also UCS parameter on DD statement)
(see also 3800 Printing Subsystem)
priority
aging (JES3) 66
automatic priority group 44,68
definition 367
dispatching (see DPRTY parameter on EXEC statement)
output (see PRTY parameter on FORMAT PR statement (JES3))
selection of jobs (see PRIORITY statement (JES2) and PRTY parameter on JOB statement (for JES3))
PRIORITY statement (JES2)
description 285
introduction 4
job selection 44
private catalogs
(see JOBCAT and STEPCAT DD statements)

private library
adding members to 90
concatenating 92
creating 89-90
defining JES3 catalog procedure 316,318
definition 367
retrieving 91
with PGM parameter 45-46
using 89
(see also JOBLIB DD and STEPLIB DD statements)
PRIVATE subparameter of VOLUME parameter
description 262,263
use of 12
private volume 367
(see also PRIVATE subparameter of VOLUME parameter)
PROC parameter
on MAIN statement (JES3) 312,313,316,318
on EXEC statement 168
PROC statement
definition 367
description 275-276
introduction 1
symbolic parameter 118-122
use of 110-112
procedure end
(see PEND statement)
procedure library (SYS1.PROCLIB)
definition 6,367
cataloged procedure 6,110-118
procedure statement 367
(see also PROC statement)
procedure step 110,367
PROCESS statement (JES3)
description 326-327
use of 5
PROCLIB parameter
on JOBPARM statement (JES2) 279
program, calling 45-46
program name subparameter of SYSOUT parameter 254
programmer's name parameter on JOB statement 141
PROTECT parameter on DD statement
description 246
PRTSP subparameter of DCB parameter 214
PRTY parameter on FORMAT PR statement (JES3) 303,306
PRTY parameter on JOB statement
description (JES2) 142
description (JES3) 143
selection of jobs (JES3) 66
PU parameter on FORMAT statement (JES3) 314,324-325
public volume
definition 366
requesting 12
(see also VOLUME parameter on DD statement)
punch
on OUTPUT statement (JES2) 282
on ROUTE statement (JES2) 286
on ROUTE statement (JES2) 342
punching a data set (JES3) (see FORMAT PU statement)
PURGE statement (JES2) 277
P11 character set (2311) 57,84

QISAM subparameters of DCB parameter 206-216
(see also indexed sequential data set)
QN character set (1403) 57,84
QNAME parameter on DD statement 247
QSAM subparameters of DCB parameter 206-216
qualified data set name 30,368
queue 368
queued indexed sequential access method
(see QISAM subparameters of DCB parameter)
queued sequential access method
(see QSAM subparameters of DCB parameter)

R subparameter
of ROUTE statement (JES2) 286
R subparameter of RD parameter
on EXEC statement 169,170
on JOB statement 144
RACF (see GROUP, PASSWORD, and USER parameters on the JOB statement and PROTECT parameter on the DD statement)
RACF protection
for direct access data sets 246
for tape volumes 246
RCK subparameter of AMP parameter 193
RD parameter
on EXEC statement 169,170
on JOB statement 144
use of (JES2) 48
use of (JES3) 76
RDJFCB macro instruction 107
reader/interpreter 368
reading a data set
dummy 94
multivolume 13
shared control 41
READ/WRITE macros before a CHECK macro
(see NCP subparameter of DCB parameter)
REAL subparameter of ADDRSPC parameter
on EXEC statement 157
on JOB statement 132
requesting storage 9-10
real storage 368
(see also REGION parameter)
RECFM subparameter
of AMP parameter 193,194
of DCB parameter 214-215
record 368
record format
(see RECFM subparameter of DCB parameter)
record key position
(see RKP subparameter of DCB parameter)
record length
(see LRECL subparameter of DCB parameter)
REF subparameter of VOLUME parameter
description 262,264-265
specific volume request 11-13
volume affinity 16-17
references, backward
(see backward references)
region 368
REGION parameter
on EXEC statement 171
on JOB statement 146
requesting storage 9-10
region request, example 10
region size, default 146,171
relational operators on the COND parameter
on EXEC statement 158
on JOB statement 134
use of (JES2) 48
use of (JES3) 73-74
relative generation number 106
relative track number
(see address subparameter of SPACE parameter)
RELEASE operator command 266
RELEASE parameter on NET statement (JES3)
description 320,322

use of 68-72
releasing space
deleting a data set (see DELETE subparameter of DISP parameter)
unused space (see RLSE subparameter of SPACE parameter)
RELSCHCT parameter on NET statement (JES3)
description 320,321,322
use of 68-72
remote devices 368
remote job entry 368
remote job processing (JES3)
definition 368
description 87
REMOTE parameter on SIGNON statement
for (JES2) 290
for (JES2) 346
remote station 368
remote terminal
on OUTPUT statement (JES2) 283
on ROUTE statement (JES2) 286
removable volume (see VOLUME parameter on DD statement)
replacing variable data
(see MODIFY parameter)
REPLY operator command 266
repositioning, tape
(see REPOS subparameter of DCB parameter)
requesting forms overflow and printer spacing 85
requesting storage (see REGION parameter)
requesting units and volumes 10
RESERVE subparameter of DCB parameter 215
reserved volumes
definition 368
private volume not demounted 12
RESET operator command 266
resources
definition 368
dynamic allocation of 25
requesting 9-42
restart
definition 368
for generation data group 108
for JES2 48
for JES3 76
(see also RD parameter, RESTART parameter on JOB statement, and SYSCHK and SYSCKEOV DD statements)
restart facility (see checkpoint/restart facility)
RESTART operator command (JES3) 292
RESTART parameter on JOB statement
description 147-148
use of (JES2) 48
use of (JES3) 76
(see also SYSCHK DD statement)
RESTART parameter
on JOBPARM statement (JES2) 279
RETAIN subparameter of VOLUME parameter
description 262,263
use of 12
retaining tape volumes
(see RETAIN subparameter of VOLUME parameter)
retention period
for DELETE 37
for KEEP 37-38
RETPD subparameter of LABEL parameter
description 238,239,240
use of 34-35
return code
definition 368
conditional execution (JES2) 47-48
conditional execution (JES3) 73
return code tests 48
rewinding tapes
for DELETE 36-37
for KEEP 37-38
RINGCHK parameter on MAIN statement (JES3) 312,313,316
RKP subparameter of DCB parameter 216
RLSE subparameter of SPACE parameter 248,250
RM subparameter
of OUTPUT statement (JES2) 283
of ROUTE statement (JES2) 286
RMT subparameter
of OUTPUT statement (JES2) 283
of ROUTE statement (JES2) 286
RN character set (1403) 57,84
RNC subparameter of RD parameter
on EXEC statement 169
on JOB statement 144
ROOM parameter
on JOBPARM statement (JES2) 279
ROUND subparameter of SPACE parameter
description 248
requesting space 20
ROUTE statement (JES2)
description 286-287
introduction 4
routing output 58-59
routing a job (JES2)
ROUTE statement (JES3) **(5740-XYN)**
description 328
introduction 5
routing in a network 62-62.1
conditional execution 47
delaying job initiation 46-47
example of 49
in a network 45
job scheduling 43-45
passing information 45
restarting 48
routing a job (JES3)
allocating data resources 63-65
conditional execution 73
delaying job initiation 81-82
example of 77
in a network 62-62.1
JES3 setup 63-65
job scheduling 61-62
passing information 74
restarting 76
selecting a job 66
selecting a processor 62
routing a job in a network (JES2)
(see also routing a job (JES2))
routing a job in a network (JES3) **(5740-XYN)**
see also routing a job (JES3))
routing output data sets (JES2) 58-59
routing output data sets (JES3) 86-87
routing system messages (JES2) 51
routing system messages (JES3) 79
routing TSO output (JES3) 87

scheduling jobs (see job scheduling)
scheduling priority 368
(see also priority)
SCAN statement (JES2) 277
SCAN subparameter of TYPRUN parameter 151
scanning JCL for syntax errors 151
scanning job without execution 75
scratch volume

(see storage volume)
SDGxx parameter on NET statement 320,321
secondary quantity subparameter of SPACE parameter
description 263,249-250
requesting space 20-23
secondary request for space
(see secondary quantity subparameter of SPACE parameter)
selecting jobs (JES2) 43-44
selecting jobs (JES3) 66-73
selecting a processor (JES3) 62
(see also SYSTEM and TYPE parameters on MAIN statement)
SEND operator command
for JCL 266
for JES3 292
separating groups of data 191,201
SEQUENCE macro
(see RESERVE subparameter of DCB parameter)
sequence of DD statements
concatenated data sets 92
sequence number subparameter of LABEL parameter
definition 238-239
use of 32
sequential data set 368
SER subparameter of VOLUME parameter
description 262,264
specific volume request 11
SET operator command 266
SETDMN operator command 266
setup of devices (JES3)
description 63-65
with conditional execution 73
SETUP parameter on MAIN statement (JES3)
description 312,313-314,317
use of 63
SETUP statement (JES2)
delaying job initiation 46-47
description 288
introduction 4
seven track tape
(see TRTCH subparameter of DCB parameter)
sharing cylinders 20
sharing a data set
(see SHR subparameter of DISP parameter)
sharing a library 91
sharing units
(see unit affinity)
sharing volumes
(see volume affinity)
SHR subparameter of DISP parameter
data set status 35-36
description 221,222-223
shared control 41
SIGNOFF statement (JES2) 289
SIGNON statement (JES2) 290
SL subparameter of LABEL parameter 238,239
SLIP operator command 266
SN character set (1403) 57,84
space
for directory 22
for index 22
for non-VSAM data sets on mass storage volumes 25
for partitioned data sets 22
primary requests 20-21
secondary requests 21
specific tracks 22
SPACE parameter on DD statement
creating a private library 89-90
description 248-251
for non-VSAM data sets 20-23
for generation data groups 106
for indexed sequential data set 102
index subparameters 22
requesting space 20-23
storing a dump 185
using mass storage volumes 24-25,264
SPART parameter **(5740-XYN)**
on JES3 MAIN statement 312,316,319
examples of using 63.0-63.1
introduction 5
(see also spool partitioning)
space on a printer
(see PRTSP subparameter of DCB parameter)
special characters 127-128
special character set (see special characters)
special data sets
dummy 93
in the input stream 96
ISAM 100-104
generation data groups (GDG) 105-109
private and temporary libraries 89-92
subsystem data set 109
virtual input/output (VIO) 94
VSAM 96-99
specific image
(see FCB parameter)
specific tracks
for partitioned data set 22
how to request 20-22
restrictions 22
specific volume request
definition 368
for mass storage volumes 25
restriction using multi-device type VSAM data set 97
system action 11,21
specifying device for an output data set (JES2) 53
specifying device for an output data set (JES3) 63-65
SPLIT parameter on DD statement
value converted to space request 111,20
spooled data set 368
spool partitioning 63-63.1
(see also SPART parameter)
STACK subparameter of DCB parameter 216
stacker bin
(see STACK subparameter of DCB parameter)
STACKER parameter on FORMAT PR statement (JES3) 86,303,305
stackers for printed output
(see BURST and STACKER parameters)
standard job 5,368
standard labels (see SL subparameter of LABEL parameter)
START operator command
for JCL 266
for JES3 292
status, data set 36
STD1, standard FCB image 57-58,85,232
STD2, standard FCB image 57-58,85,232
STD3, standard FCB image 57-58,85,232
STDWTR on SYSOUT DD statement 254
step restart
definition 368
for generation data group 108
use of (JES2) 48
use of (JES3) 76
(see also RESTART parameter on JOB statement and RD parameter)
STEPCAT DD statement
description 181
master catalog 92
private catalog 92
VSAM 96-99

STEPLIB DD statement
  defining and retrieving private library 89-91
  concatenating private libraries 92
  description 182
  effect with JOBLIB DD statement 90-92
stepname
  assigning 155
  definition 368
STOP operator command 266
STOPMN operator command 266
storage (see REGION parameter)
storage volume 368
STRNO subparameter of AMP parameter 193,104
SUBALLOC parameter on DD statement
  value converted to space request iii,20
subparameter, definition 125
subsystem definition 368
SUBSYS parameter on the DD statement
  description 252
  effect on DD statement parameters 109,252
  subparameters, specifying 252
subsystem data sets
  creating and using 109,252
successor job
  dependent job control 68-72
  early setup of 68
SUL subparameter of LABEL parameter 238,239-240
suppressed processing, data set 93
suppressed writing, data set (JES2) 54
suppressed writing, data set (JES3) 82
SWITCH operator command (JES3) 292
symbolic parameters
  assigning values to 118-122
  default values 118
  definition 368
  example of 122
  nullifying 120
  use of commas 120-121
  with PROC statement 168
SYNAD subparameter of AMP parameter 193,195
syntax
  checking 46,151
  for JES2 277
  for JES3 291
SYSABEND DD statement
  description 185-186
  requesting abnormal termination dumps (JES2) 51-52
  requesting abnormal termination dumps (JES3) 79-80
SYSAFF parameter
  on JOBPARM statement (JES2) 279
SYSCHK DD statement
  description 188
  use of (JES2) 48
  use of (JES3) 76
  (see also RESTART parameter on JOB statement)
SYSCKEOV DD statement 190
SYSMDUMP DD statement
  description 185-186
  requesting abnormal termination dumps (JES2) 51-52
  requesting abnormal termination dumps (JES3) 79-80
SYSOUT parameter on DD statement
  description 254
  multiple copies of output 55,83
  print a dump 51-52,79-80
system affinities (JES2)
  (see SYSAFF parameter on JOBPARM statement)
system generation 368
system library (SYS1.LINKLIB) 368
system messages 368
  (see also MSGLEVEL parameter on JOB statement)
system output device 51,79,368
SYSTEM parameter on MAIN statement (JES3)
  description 312,313,317
  use of 62
system resources manager 368
SYSUDUMP DD statement
  description 185-186
  requesting abnormal termination dump (JES2) 51-52
  requesting abnormal termination dump (JES3) 79-80
SYS1.LINKLIB data set
  (see system library)
SYS1.PROCLIB data set
  (see procedure library)

table name subparameter of CHARS parameter
  on DD statement 197
  on FORMAT PR statement (JES3) 303,305,307
table reference character 57,84,210,368
  (see also trc subparameter of MODIFY)
tape density
  (see DEN subparameter of DCB parameter)
tape device
  eligible for allocation 12
tape labels, ANSI
  (see LABEL parameter on DD statement)
tapemark 32-34
task 368
TCAM
  NOTIFY parameter 138
  QNAME parameter 247
  TERM parameter 256
  TCAM subparameters of DCB parameter 206-216
tcamname subparameter of QNAME parameter 247
telecommunication access method
  (see TCAM subparameters of DCB parameter)
teleprocessing device 10-11
temporary data set
  creating 29-30
  definition of 29,369
  deletion, conditions of 30
  DSNAME parameter, use of 31-32
  member name 30
  specifying disposition 221
  VIO, use of 94-95
temporary library
  definition of 92,369
  with PGM parameter 166
TERM parameter on DD statement 256
testing JCL without execution 75
  (see also PGM parameter on EXEC statement)
THRESH subparameter of DCB parameter 216
THRESHLD subparameter on FORMAT PR statement 303,305
TIME parameter
  on EXEC statement 172-173
  on JOB statement 149-150
  on JOBPARM statement (JES2) 279,280
time-dependent program
  requesting real storage 9
time limit for CPU
  on EXEC statement 172
  on JOB statement 149
TN character set (1403) 57,84
TRACE subparameter of AMP parameter 193,195
track number, relative
  (see address subparameter of SPACE parameter)
tracks
  capacities 358
  space requests 20-23
  description 249,369
  cylinder index
    (see NTM subparameter of DCB parameter)

overflow
(see CYLOFL subparameter of DCB parameter)
searching
(see LIMCT subparameter of DCB parameter)
TRAIN parameter on FORMAT PR statement (JES3) 303,305,307
transmitting data to other nodes (JES2) **(5740-XYS)** 45
trc subparameter of MODIFY parameter
on DD statement 242
on FORMAT PR statement (JES3) 303,306,307
on OUTPUT statement (JES2) **(5740-XYS)** 282,284,284.2
TRK subparameter of SPACE parameter
space requests 20
description 248
TRTCH subparameter of DCB parameter 216
TYPE parameter on MAIN statement (JES3)
description 312,313,314,317
defining control program 62
TYPRUN parameter on JOB statement
delaying job initiation (JES2) 46-47
delaying job initiation (JES3) 67
description 151-152
T11 character set (3211) 57,84

U subparameter
of OUTPUT statement (JES2) 283
of ROUTE statement (JES2) 286
UCS (universal character set)
(see UCS parameter)
UCS parameter
on DD statement 257
on OUTPUT statement (JES2) 282,283
use of (JES2) 56-57
use of (JES3) 84
uncataloging a data set
(see UNCATLG subparameter of DISP parameter)
UNCATLG subparameter of DISP parameter
description 221,222,223
use of 38
unit
definition 10,369
information from other than UNIT parameter 15
specifying output device 52
type of unit you can specify 13-14
(see also UNIT parameter on DD statement)
unit address subparameter of UNIT parameter
definition 370
description 259,260
use of 13,14
unit affinity 369
(see also AFF subparameter of UNIT parameter)
unit count subparameter of UNIT parameter 259,260
unit information, other than UNIT parameter 15
unit of measurement
cylinders, blocks and tracks 20-22
unit override 261
UNIT parameter on DD statement
creating a private library 89-90
description 259-261
generation data groups 106,108
indexed sequential data sets 101,103
multiple units 13-14
requesting units 13-15
unit affinity 15-17
storing a dump 51-52,79-80
using mass storage volumes 24-25
write output data set 52,80
unit record device 10
universal character set 369
(see also UCS parameter)
unlabeled tapes (see LABEL parameter on DD statement)
UNLOAD operator command 266
UPDATE parameter on MAIN statement (JES3)
description 312,316,319
use of 111
USER parameter
on FORMAT AC statement (JES3) 299,300
on JOB statement 153
on MAIN statement (JES3) 312,316
user-assigned group name subparameter of UNIT parameter
definition 369
description 259,260
use of 13
user labels (see LABEL parameter on DD statement)
userid subparameter of DEST parameter
for TSO 219,220
for VM 219,220
restrictions when coding 220
(see also SYSOUT parameter)
userid subparameter of NOTIFY statement (JES2) 336
userid subparameter of USER parameter 153

V format
(see RECFM subparameter of DCB parameter)
V=R dynamic area
(see nonpageable dynamic area)
V=V
(see virtual storage)
VARY operator command
for JCL 266
for JES3 292
VERIFY subparameter
of FCB parameter 232
of UCS parameter 257
VIO (see virtual input/output)
VIRT subparameter of ADDRSPC parameter
on EXEC statement 157
on JOB statement 132
requesting storage 9-10
virtual direct access device
(see mass storage volumes)
virtual input/output (VIO)
definition 369
for temporary data sets 94-95
virtual storage 369
(see also REGION parameter)
virtual storage access method (VSAM) 96-99
virtual volumes
mass storage system 24-25
(see also MSVGP parameter on DD statement)
VOL parameter on DD statement
(see VOLUME parameter on DD statement)
volume
deferred mounting 14
definition 369
maximum volume request 13
(see also VOLUME parameter on DD statement)
volume affinity
definition 13,369
how to request 13,16-17
volume count subparameter of VOLUME parameter
maximum number of volumes you can request 13
description 262,263
VOLUME parameter on DD statement
creating a private library 89-90
demounting of volumes 12
description 262-265
generation data groups 106,262
ISAM requirement for 101,104
nonspecific volume request 11-12
for multivolume data set 13
for private volumes 12
for volume affinity 13-14,16-17
specific volume request 11

using mass storage volumes 24-25
volume sequence number subparameter of VOLUME parameter
description 262,263
use of 13
volume serial number
subparameter of VOLUME parameter 262,264
parameter on SETUP statement (JES2) 288
volumes, requesting
(see VOLUME parameter)
volume table of contents 369
VSAM (virtual storage access method)
for private catalogs 92
processing VSAM data sets 96-99

wait-state time limit
(see TIME parameter on JOB and EXEC statement)
work station, controlling output to 58-59,86-87,369
working set 369
WRITELOG operator command 266
writing a dummy data set 94
writing output data sets
for JES2 52-58
for JES3 80-87

XEQ parameter on ROUTE statement (JES2) **(5740-XYS)** 342
XEQ parameter on ROUTE statement (JES3) **5740-XYN)** 328
XEQ statement (JES2)
description 347
example 347
introduction 4
XMIT statement (JES2) **(5740-XYS)** 290.2
XN character set (1403) 57,84
XX (cataloged procedure) 117
XX* (cataloged procedure) 117
X/ (cataloged procedure) 117

Y subparameter of BURST parameter
on DD statement 196
on OUTPUT statement (JES2) 282
YES subparameter of DD HOLD parameter 237
YN character set (1403) 57,84

Z parameter
on OUTPUT statement (JES2) 282

6, FCB image 57-58
8, FCB image 57-58
1403 printer
requesting a special character set 56-57,84
requesting specific forms control 57-58,85
1440
(see TIME parameters on JOB and EXEC statements)
3211 printer
requesting a special character set 56-57,84
requesting specific forms control 57-58,85
3330 model 11
coding 260
track capacities 358
3330V virtual volume 24,243
3340 fixed head feature 14
3348 model 70F data module 14
3525 punch interpretation 86
3540 diskette reader writer 31,191,201
3800 Printing Subsystem
bursting the output (see BURST and STACKER parameters)
character arrangement tables (see CHARS parameter)
copy modification module (see MODIFY parameter)
forms control 57-58,85
modifying a data set (see MODIFY parameter)
output for JES2 (see OUTPUT statement)
output for (JES3) (see FORMAT PR statement)
printing a dump with more data per page 197,232
printing multiple copies 55,83
printing with forms overlay (see FLASH parameter)
stacking the paper output (see BURST and STACKER parameters)
3850 Mass Storage System
(see MSVGP parameter on DD statement)

tape device
eligible for allocation 12
tape labels, ANSI
(see LABEL parameter on DD statement)
tapemark 32-34
task 368
TCAM
NOTIFY parameter 138
QNAME parameter 247
TERM parameter 256
TCAM subparameters of DCB parameter 206-216
telecommunication access method
(see TCAM subparameters of DCB parameter)
teleprocessing device 10-11
temporary data set
creating 29-30
definition of 29,369
deletion, conditions of 30
DSNAME parameter, use of 31-32
member name 30
specifying disposition 221
VIO, use of 94-95
temporary library
definition of 92,369
with PGM parameter 166
TERM parameter on DD statement 256
testing JCL without execution 75
(see also PGM parameter on EXEC statement)
THRESH subparameter of DCB parameter 216
THRESHLD subparameter on FORMAT PR statement 303,305
TIME parameter
on EXEC statement 172-173
on JOB statement 149-150
on JOBPARM statement (JES2) 279,280
on JOBPARM statement (JES2 NJE) 331
time-dependent program
requesting real storage 9
time limit for CPU
on EXEC statement 172
on JOB statement 149
TN character set (1403) 57,84
TRACE subparameter of AMP parameter 193,195
track number, relative
(see address subparameter of SPACE parameter)
tracks
capacities 358
space requests 20-23
description 249,369
cylinder index
(see NTM subparameter of DCB parameter)
overflow
(see CYLOFL subparameter of DCB parameter)
searching
(see LIMCT subparameter of DCB parameter)
TRAIN parameter on FORMAT PR statement (JES3) 303,305,307
trc subparameter of MODIFY parameter
on DD statement 242
on FORMAT PR statement (JES3) 303,306,307
TRK subparameter of SPACE parameter
space requests 20
description 248
TRTCH subparameter of DCB parameter 216
TSO on an ASP main processor 87
(see also FORMAT AC and MAIN statements (JES3))
TYPE parameter on MAIN statement (JES3)
description 312,313,314,317
defining control program 62
TYPRUN parameter on JOB statement
delaying job initiation (JES2 or JES2 NJE) 46-47
delaying job initiation (JES3) 67
description 151-152
T11 character set (3211) 57,84

U subparameter
of OUTPUT statement (JES2) 283
of OUTPUT statement (JES2 NJE) 337
of ROUTE statement (JES2) 286
of ROUTE statement (JES2 NJE) 342
UCS (universal character set)
(see UCS parameter)
UCS parameter
on DD statement 257
on OUTPUT statement (JES2) 282,283
on OUTPUT statement (JES2 NJE) 337
use of (JES2 or JES2 NJE) 56-57
use of (JES3) 84
uncataloging a data set
(see UNCATLG subparameter of DISP parameter)
UNCATLG subparameter of DISP parameter
description 221,222,223
use of 38
unit
definition 10,369
information from other than UNIT parameter 15
specifying output device 52
type of unit you can specify 13-14
(see also UNIT parameter on DD statement)
unit address subparameter of UNIT parameter
definition 370
description 259,260
use of 13,14
unit affinity 369
(see also AFF subparameter of UNIT parameter)
unit count subparameter of UNIT parameter 259,260
unit information, other than UNIT parameter 15
unit of measurement
cylinders, blocks and tracks 20-22
unit override 261
UNIT parameter on DD statement
creating a private library 89-90
description 259-261
generation data groups 106,108
indexed sequential data sets 101,103
multiple units 13-14
requesting units 13-15
unit affinity 15-17
storing a dump 51-52,79-80
using mass storage volumes 24-25
write output data set 52,80
unit record device 10
universal character set 369
(see also UCS parameter)
unlabeled tapes (see LABEL parameter on DD statement)
UNLOAD operator command 266

UPDATE parameter on MAIN statement (JES3)
description 312,316,319
use of 111
USER parameter
on FORMAT AC statement (JES3) 299,300
on JOB statement 153
on MAIN statement (JES3) 312,316
user-assigned group name subparameter of UNIT parameter
definition 369
description 259,260
use of 13
user labels (see LABEL parameter on DD statement)
userid subparameter of NOTIFY statement (JES2 NJE) 336
userid subparameter of USER parameter 153

V format
(see RECFM subparameter of DCB parameter)
V=R dynamic area
(see nonpageable dynamic area)
V=V
(see virtual storage)
VARY operator command
for JCL 266
for JES3 292
VERIFY subparameter
of FCB parameter 232
of UCS parameter 257
VIO (see virtual input/output)
VIRT subparameter of ADDRSPC parameter
on EXEC statement 157
on JOB statement 132
requesting storage 9-10
virtual direct access device
(see mass storage volumes)
virtual input/output (VIO)
definition 369
for temporary data sets 94-95
virtual storage 369
(see also REGION parameter)
virtual storage access method (VSAM) 96-99
virtual volumes
mass storage system 24-25
(see also MSVGP parameter on DD statement)
VOL parameter on DD statement
(see VOLUME parameter on DD statement)
volume
deferred mounting 14
definition 369
maximum volume request 13
(see also VOLUME parameter on DD statement)
volume affinity
definition 13,369
how to request 13,16-17
volume count subparameter of VOLUME parameter
maximum number of volumes you can request 13
description 262,263
VOLUME parameter on DD statement
creating a private library 89-90
demounting of volumes 12
description 262-265
generation data groups 106,262
ISAM requirement for 101,104
nonspecific volume request 11-12
for multivolume data set 13
for private volumes 12
for volume affinity 13-14,16-17
specific volume request 11
using mass storage volumes 24-25
volume sequence number subparameter of VOLUME parameter
description 262,263
use of 13
volume serial number
subparameter of VOLUME parameter 262,264
parameter on SETUP statement (JES2) 288
parameter on SETUP statement (JES2 NJE) 344
volumes, requesting
(see VOLUME parameter)
volume table of contents 369
VSAM (virtual storage access method)
for private catalogs 92
processing VSAM data sets 96-99

wait-state time limit
(see TIME parameter on JOB and EXEC statement)
work station, controlling output to 58-59,86-87,369
working set 369
WRITELOG operator command 266
writing a dummy data set 94
writing output data sets
for JES2 or JES2 NJE 52-58
for JES3 80-87

XEQ parameter on ROUTE statement (JES2 NJE) 342
XEQ statement (JES2 NJE)
description 347
example 347
introduction 4
XN character set (1403) 57,84
XX (cataloged procedure) 117
XX* (cataloged procedure) 117
X/ (cataloged procedure) 117

Y subparameter of BURST parameter
on DD statement 196
on OUTPUT statement (JES2) 282
on OUTPUT statement (JES2 NJE) 337
YES subparameter of DD HOLD parameter 237
YN character set (1403) 57,84

Z parameter
on OUTPUT statement (JES2) 282
on OUTPUT statement (JES2 NJE) 337

6, FCB image 57-58
8, FCB image 57-58
1403 printer
requesting a special character set 56-57,84
requesting specific forms control 57-58,85
1440
(see TIME parameters on JOB and EXEC statements)

3211 printer
requesting a special character set 56-57,84
requesting specific forms control 57-58,85
3330 model 11
coding 260
track capacities 358
3330V virtual volume 24,243
3340 fixed head feature 14
3348 model 70F data module 14
3525 punch interpretation 86
3540 diskette reader writer 31,191,201
3800 Printing Subsystem
bursting the output (see BURST and STACKER parameters)
character arrangement tables (see CHARS parameter)
copy modification module (see MODIFY parameter)
forms control 57-58,85
modifying a data set (see MODIFY parameter)
output for (JES2 or JES2 NJE) (see OUTPUT statement)
output for (JES3) (see FORMAT PR statement)
printing a dump with more data per page 197,232
printing multiple copies 55,83
printing with forms overlay (see FLASH parameter)
stacking the paper output (see BURST and STACKER parameters)
3850 Mass Storage System
(see MSVGP parameter on DD statement)

OS/VS2 MVS JCL
GC28-0692-5

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate.

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

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 comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If comments apply to a Selectable Unit, please provide the name of the Selectable Unit _____.

If you wish a reply, give your name and mailing address:

_____

_____

_____

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

Please circle the description that most closely describes your occupation.

Customer	(Q) Install Mgr.	(U) System Consult.	(X) System Analyst	(Y) System Prog.	(Z) Applica. Prog.	(F) System Oper.	(I) I/O Oper.	(L) Term. Oper.

(O) Other

IBM	(S) System Eng.	(P) Prog. Sys. Rep.	(A) System Analyst	(B) System Prog.	(C) Applica. Prog.	(D) Dev. Prog.	(R) Comp. Prog.	(G) System Oper.	(J) I/O Oper.	(E) Ed. Dev. Rep.	(N) Cust. Eng.	(T) Tech. Staff Rep.

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.)

GC28-0692-5

Reader's Comment Form

Cut or Fold Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

OS/VS2 MVS JCL (S370-36) Printed in U.S.A. GC28-0692-5