

IBM z/VSE



# System Control Statements

*Version 3 Release 1*



IBM z/VSE



# System Control Statements

*Version 3 Release 1*

**Note!**

Before using this information and the product it supports, be sure to read the general information under "Notices" on page xi.

**First Edition (March 2005)**

This edition applies to Version 3 Release 1 of IBM z/Virtual Storage Extended (z/VSE), Program Number 5609-ZVS,

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the addresses given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Entwicklung GmbH  
Department 3248  
Schoenaicher Strasse 220  
D-71032 Boeblingen  
Federal Republic of Germany

You may also send your comments by FAX or via the Internet:

Internet: [s390id@de.ibm.com](mailto:s390id@de.ibm.com)  
FAX (Germany): 07031-16-3456  
FAX (other countries): (+49)+7031-16-3456

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1984, 2005. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

**Figures . . . . . vii**

**Tables . . . . . ix**

**Notices . . . . . xi**

Trademarks and Service Marks . . . . . xi

**About This Book . . . . . xiii**

Who Should Use This Book. . . . . xiii

How to Use This Book . . . . . xiii

Where to Find More Information . . . . . xiv

**Summary of Changes . . . . . xv**

**Chapter 1. Introduction . . . . . 1**

Initial Program Load. . . . . 1

Job Control . . . . . 1

Attention Routine. . . . . 1

Linkage Editor. . . . . 1

Librarian . . . . . 1

Edited Macro Service Program (ESERV) . . . . . 2

System Buffer Load (SYSBUFLD) . . . . . 2

Maintain System History Program (MSHP) . . . . . 2

Understanding Syntax Diagrams. . . . . 2

    Frequent Abbreviations . . . . . 4

    Continuation of Commands and Statements . . . . . 5

**Chapter 2. Initial Program Load . . . . . 7**

The IPL Load Parameter . . . . . 9

The Supervisor Parameters Command . . . . . 11

ADD . . . . . 13

    Format 1 . . . . . 13

    Format 2 . . . . . 14

    Format 3 . . . . . 14

    Format 4 . . . . . 15

DEF . . . . . 16

DEF SCSI . . . . . 17

DEL . . . . . 18

DEV . . . . . 19

DLA . . . . . 20

DLF . . . . . 22

DPD . . . . . 24

SET . . . . . 26

SET XPCC . . . . . 28

SET ZONEDEF / SET ZONEBDY . . . . . 29

    SET ZONEDEF . . . . . 29

    SET ZONEBDY . . . . . 30

SVA . . . . . 32

SYS . . . . . 34

Storage Allocation Rules . . . . . 38

Device Type Codes . . . . . 40

**Chapter 3. Job Control and Attention**

**Routine . . . . . 45**

Job Control Statements . . . . . 48

Job Control and Attention Routine Commands . . . . 48

Job Control Statements Summary . . . . . 49

Sequence of JCS and JCC . . . . . 50

Conditional Job Control . . . . . 50

Parameterized Procedures . . . . . 51

Symbolic Parameters . . . . . 51

Nested Procedures . . . . . 53

Scope of Symbolic Parameters . . . . . 53

Printing of Job Control Statements and Commands . 54

Command Authorization in Job Control . . . . . 54

Command Authorization in Attention Routine (AR) . 55

Command and Statement Formats (JCS, JCC, AR) . 55

    ALLOC (Allocate Storage to Partitions) . . . . . 56

    ALTER (Alter Contents of Virtual Storage) . . . . 59

    ASSGN (Assign Logical Unit) . . . . . 60

    BANDID (Mount or Query 4248 Print Band) . . . . 70

    BATCH (Start or Continue Processing) . . . . . 71

    CACHE (Control Cache Operations) . . . . . 72

    CANCEL (Cancel Job or I/O Request) . . . . . 80

    CLOSE (Close Output Logical Unit) . . . . . 82

    DATE (Override System Date) . . . . . 85

    DLBL (Disk Label Information) . . . . . 86

    DSPLY (Display Virtual Storage) . . . . . 91

    DUMP (Dump Storage Areas) . . . . . 92

    DVCNDN (Device Down) . . . . . 95

    DVCUP (Device Up) . . . . . 96

    END or ENTER (End of Input) . . . . . 97

    EXEC (Execute Program or Procedure) . . . . . 98

    EXPLAIN (Online Explanation Support) . . . . . 105

    EXTENT (Disk or Diskette Extent Information) . 106

    FREE (Reset RESERV Command) . . . . . 110

    GETVIS (Display GETVIS Information) . . . . . 111

    GOTO (Skip to Label) . . . . . 113

    HCLOG (Control Message Logging) . . . . . 114

    HOLD (Hold Assignments and LIBDEFs) . . . . . 115

    ID (User-ID and Password) . . . . . 116

    IF (Check Local Condition) . . . . . 117

    IGNORE (Ignore Abnormal Condition) . . . . . 119

    IXFP SNAP . . . . . 120

    JCLEXIT (Multiple JCL Exits) . . . . . 123

    JOB (Identify Job) . . . . . 124

    LFCB (Load Forms Control Buffer) . . . . . 125

    LIBDEF (Define Sublibrary Chain) . . . . . 126

    LIBDROP (Drop Sublibrary Chain) . . . . . 129

    LIBLIST (Query Sublibrary Chains) . . . . . 130

    LIBSERV (Control IBM 3494 Tape Library

        Dataserver) . . . . . 131

    LISTIO (Query I/O Assignments) . . . . . 141

    LOG (Log JC Statements) . . . . . 143

    LUCB (Load Universal Character-Set Buffer) . . . 145

    MAP (Display Storage Layout) . . . . . 146

    MSECS (Change or Query Time Slice) . . . . . 153

MSG (Communicate With Program) . . . . .	154
MTC (Magnetic Tape Control) . . . . .	155
NEWVOL (Alter Volume Assignment) . . . . .	157
NOLOG (Suppress JC Logging) . . . . .	158
NPGR (Number of Programmer Logical Units)	159
OFFLINE (Simulate DEVICE OR CHPID NOT READY) . . . . .	160
ON (Set Global Condition) . . . . .	161
ONLINE (Simulate DEVICE OR CHPID READY) . . . . .	164
OPERATE (Query or Alter Mode of Operation and Console State) . . . . .	165
OPTION (Set Temporary JC Options) . . . . .	167
PAUSE (Suspend Processing) . . . . .	176
PROC (Procedure) . . . . .	177
PRTY (Query and Set Partition Priorities) . . . . .	178
PRTY SHARE Command . . . . .	181
PRTYIO (Query and Set Partition Priorities for I/O Requests) . . . . .	182
PWR (Pass POWER Command) . . . . .	184
PWROFF (Power Off CPU) . . . . .	185
QUERY (Query Data Spaces and Standard Options) . . . . .	186
QUERY SCSI . . . . .	191
QUERY TD . . . . .	192
RC (Request Communication) . . . . .	194
REDISPLAY (Retrieve Logged Information) . . . . .	195
REPLID (Query Reply-IDs) . . . . .	199
RESERV (Reserve Device for VSE/VSAM) . . . . .	200
RESET (Reset ASSGNs and LIBDEFs to Permanent Values) . . . . .	201
ROD (Record on Demand) . . . . .	202
RSTRT (Restart Checkpointed Program) . . . . .	203
SET (Set Program Control Values) . . . . .	204
SETDF (Set 3800 Printer Defaults) . . . . .	208
SETPARM (Set Symbolic Parameter) . . . . .	210
SETPFIX (Set PFI Limits) . . . . .	212
SETPRT (Set 3800 Printer Values for Job) . . . . .	214
SIZE (Program Size) . . . . .	219
START (Start or Continue Processing) . . . . .	221
STATUS (Display Task or Device Status) . . . . .	222
STDOPT (Standard JC Options) . . . . .	225
STOP (Stop Processing) . . . . .	228
SYSDEF (Define Data Space) . . . . .	229
SYSDEF SCSI (Define SCSI Device) . . . . .	232
SYSDEF TD . . . . .	234
SYSECHO (VM as z/VSE Master Console) . . . . .	235
TLBL (Tape Label Information) . . . . .	236
TPBAL (Telecommunication Balancing) . . . . .	240
UCS (Load Universal Character-Set Buffer) . . . . .	241
UNBATCH (Deactivate Foreground Partition) . . . . .	242
UNLOCK (Release Locked Resources) . . . . .	243
UPSI (User Program Switch Indicators) . . . . .	244
VDISK (Define Virtual Disk) . . . . .	245
VOLUME (Query Mounted Volumes) . . . . .	247
VTAPE (Define/Release Virtual Tape) . . . . .	249
ZONE (Set Time Zone) . . . . .	251
/. (Label Statement) . . . . .	252
/+ (End-of-Procedure) . . . . .	253
/* (End-of-Data File) . . . . .	254
/& (End-of-Job) . . . . .	255

* CP (Submit CP Commands) . . . . .	256
* (COMMENTS) . . . . .	257
Job Control Statement Examples . . . . .	258
General Job Control Examples . . . . .	258
Conditional Job Control: Example of IF Statement . . . . .	261
Conditional Job Control: Example of ON, IF and GOTO Statements for Abnormal Termination . . . . .	262
Parameterized Procedure Example . . . . .	263
Parameterized Procedures and Procedure Nesting Example . . . . .	264

## Chapter 4. Linkage Editor . . . . . 267

YEAR 2000 Support . . . . .	268
Linkage Editor Return Codes . . . . .	269
Language Translator Modules . . . . .	269
Linkage Editor Control Statements . . . . .	270
General Control Statement Format . . . . .	270
Control Statement Placement . . . . .	270
ACTION . . . . .	273
ENTRY . . . . .	275
INCLUDE . . . . .	276
MODE . . . . .	278
PHASE . . . . .	280

## Chapter 5. Librarian . . . . . 285

Librarian Return Codes . . . . .	285
Library Concept . . . . .	285
Supported Equipment . . . . .	286
Compatibility with Previous Releases . . . . .	286
Librarian Command Syntax . . . . .	286
Invoking the Librarian Program . . . . .	288
Partition Size for the Librarian Program . . . . .	289
Summary of Librarian Commands . . . . .	289
Merging Sublibraries . . . . .	290
Librarian Commands . . . . .	290
ACCESS (Specify Target Sublibrary) . . . . .	290
BACKUP (Backup Library or Sublibrary) . . . . .	292
CATALOG (Catalog Member) . . . . .	295
CHANGE (Change REUSE Attribute) . . . . .	297
COMPARE (Compare Libraries, Sublibraries or Members) . . . . .	298
CONNECT (Specify 'From' and 'To' Sublibraries) . . . . .	300
COPY (Copy Library, Sublibrary or Member) . . . . .	301
DEFINE (Define Library or Sublibrary) . . . . .	305
DELETE (Delete Library, Sublibrary or Member) . . . . .	307
GOTO (Skip to Label) . . . . .	308
INPUT (Read from SYSIPT) . . . . .	309
LIST (List Member Contents) . . . . .	310
LISTDIR (List Directory Information) . . . . .	311
LOCK (Lock Member) . . . . .	314
MOVE (Move Library, Sublibrary or Member) . . . . .	315
ON (Set Global Condition) . . . . .	319
PUNCH (Punch Member Contents) . . . . .	321
RELEASE (Release Unused Shared Space) . . . . .	323
RENAME (Rename Sublibrary or Member) . . . . .	324
RESTORE (Restore Backed-up Library, Sublibrary or Member) . . . . .	325
SEARCH (Search for Library Member) . . . . .	330
TEST (Test Library Integrity) . . . . .	332

UNLOCK (Unlock Member) . . . . .	333
UPDATE (Alter Member Contents) . . . . .	334
UPDATE Subcommands . . . . .	336
)ADD (Add Line to Member) . . . . .	337
)DEL (Delete Line from Member) . . . . .	338
)END (Finish Update) . . . . .	339
)REP (Replace Line in Member) . . . . .	340
/. (Label Statement) . . . . .	341
/+ (End-of-DATA) . . . . .	342
/* or END (Librarian End-of-Session) . . . . .	343

**Chapter 6. Edited Macro Service Program (ESERV) . . . . . 345**

ESERV Control Statements . . . . .	346
GENCATALS (Specify Macro Output Format) . . . . .	346
DSPLY, PUNCH, DSPCH (Specify Output Destination) . . . . .	347
) ADD (Add Statement to Macro) . . . . .	348
) COL (Control Macro Statement Numbering) . . . . .	349
) DEL (Delete Statement from Macro) . . . . .	350
) END (Finish Macro Update) . . . . .	351
) REP (Replace Statement in Macro) . . . . .	352
) RST (Change Macro Statement Numbering) . . . . .	353
) VER (Verify Contents of Macro Statement) . . . . .	354

**Chapter 7. System Buffer Load (SYSBUFLD) . . . . . 355**

Control Statements . . . . .	355
BANDID . . . . .	355
FCB . . . . .	356
UCB . . . . .	356
Buffer Load Phases . . . . .	357
Standard Buffer Image Phases . . . . .	357
Additional UCB Images . . . . .	357
Automatic Buffer Loading During IPL . . . . .	359
Creating Your Own UCB/FCB Image Phases . . . . .	359
Loading the FCB Using SYSIPT . . . . .	362
FCB Characters . . . . .	362
Examples of FCB Image Phases . . . . .	363

**Chapter 8. Maintain System History Program (MSHP) . . . . . 365**

High Level Assembler for VSE . . . . .	368
Called System Control Programs . . . . .	368
Types of History Files . . . . .	368
System History File . . . . .	368
Auxiliary History File . . . . .	368
Restrictions . . . . .	369
MSHP Return Codes . . . . .	370
Repairing the History File . . . . .	370
Rules for Writing MSHP Control Statements . . . . .	371
Coding Conventions for Frequently Used MSHP Operands . . . . .	372
Function Control Statements . . . . .	374
APPLY . . . . .	374
ARCHIVE . . . . .	377
BACKUP HISTORY . . . . .	379
BACKUP PRODUCT . . . . .	380

COPY HISTORY . . . . .	382
CORRECT . . . . .	383
CREATE HISTORY . . . . .	386
DUMP HISTORY . . . . .	387
INCORPORATE . . . . .	388
INSTALL PRODUCT/SYSRES . . . . .	389
INSTALL SERVICE/BACKOUT . . . . .	393
LIST . . . . .	396
LOOKUP . . . . .	398
MERGE HISTORY . . . . .	401
PATCH . . . . .	403
PERSONALIZE . . . . .	404
REMOVE . . . . .	406
RESIDENCE . . . . .	408
RESTORE PRODUCT/SYSRES . . . . .	409
RESTORE HISTORY . . . . .	411
RETRACE . . . . .	412
REVOKE . . . . .	414
SELECT . . . . .	415
TAILOR . . . . .	416
UNDO . . . . .	419
Detail Control Statements . . . . .	420
AFFECTS . . . . .	420
ALTER . . . . .	422
COMPATIBLE . . . . .	423
COMPRISES . . . . .	424
DATA . . . . .	425
DEFINE HISTORY . . . . .	426
DELETE . . . . .	428
EXCLUDE . . . . .	429
EXECUTE . . . . .	430
GENERATE . . . . .	431
INCLUDE . . . . .	432
INFLUENCES . . . . .	433
INSERT . . . . .	434
INVOLVES . . . . .	435
OR . . . . .	436
PTF . . . . .	437
REPLACE . . . . .	438
REQUIRES . . . . .	439
RESOLVES . . . . .	441
RESTART . . . . .	442
SCAN . . . . .	443
SUPERSEDES . . . . .	444
VERIFY . . . . .	445

**Appendix. Format of Linkage Editor Statements . . . . . 447**

Format of the ESD Statement . . . . .	447
Format of the TXT Statement . . . . .	448
Format of the RLD Statement . . . . .	448
Format of the END Statement . . . . .	449
Format of the REP (User Replace) Statement . . . . .	449
External Symbol Dictionary . . . . .	450

**Glossary . . . . . 451**

**Index . . . . . 465**





---

## Figures

1. The IPL Load Parameter Format . . . . .	9	13. VOLUME Command Output Example	248
2. ESA Address Space Layout and IPL Operands Determining the Size of The Address Spaces .	38	14. General Job Control Examples Part 1	258
3. IPL Storage Values . . . . .	39	15. General Job Control Examples Part 2	259
4. Cache Statistics for 3990-3 / 3990-6. . . . .	74	16. General Job Control Examples Part 3	260
5. JCC Scenario for LIBSERV Command	139	17. General Job Control Examples Part 4	260
6. Operation Codes for MTC Statement	155	18. The Use of the IF Statement. . . . .	261
7. Example of QUERY DSPACE Output	188	19. IF, ON and GOTO Statements for Abnormal Termination . . . . .	262
8. Example of QUERY DSPACE,F3 Output	188	20. The Use of a Parameterized Procedure	263
9. Example of QUERY DSPACE,ALL Output	189	21. Use of Nested Procedures . . . . .	264
10. Example of QUERY SETPARM,PWRJOB Output. . . . .	190	22. List of Librarian Commands . . . . .	290
11. Output Example of QUERY SCSI Command	191	23. Standard Buffer Load Phases . . . . .	357
12. Output Example of QUERY TD Command	192	24. Formats of UCB/FCB Image Phases . . . . .	360
		25. Repairing the History File . . . . .	371



---

## Tables

1. Device Type Codes . . . . .	43	8. Additional UCB Images . . . . .	358
2. JCS, JCC, and AR by Function . . . . .	45	9. Function Control Statements - Overview and Purpose . . . . .	366
3. Job Control Statements Summary . . . . .	49	10. Detail Control Statements - Overview and Purpose . . . . .	367
4. Device Class Assignments. . . . .	64	11. Usage of MSHP Auxiliary History File . . . . .	369
5. Device Search Order . . . . .	66	12. Detail Control Statements Related to ARCHIVE Operands . . . . .	377
6. Mode Settings for Tapes . . . . .	66		
7. Number of Tracks per Cylinder for Disk Devices . . . . .	108		



---

## Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

Any pointers in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this publication or accessed through an IBM Web site that is mentioned in this publication.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Deutschland GmbH  
Department 0790  
Pascalstr. 100  
70569 Stuttgart  
Germany

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

---

## Trademarks and Service Marks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

Advanced Function Printing  
AFP  
CICS  
CICS/VSE  
DB2  
DFSMS/VM  
ECKD  
Enterprise Systems Architecture/370  
Enterprise Systems Architecture/390  
ES/9000  
ESCON  
IBM  
Magstar  
MVS  
NetView  
OS/390

Print Services Facility  
RAMAC  
SQL/DS  
System/370  
VM/ESA  
VSE/ESA  
VTAM  
z/OS  
z/VSE

Microsoft, Windows<sup>®</sup>, Windows NT<sup>®</sup>, and the Windows logo are trademarks of Microsoft<sup>®</sup> Corporation in the United States, other countries, or both.

---

## About This Book

**z/VSE is the successor to IBM's VSE/ESA product. Many products and functions supported on z/VSE may continue to use VSE/ESA in their names.**

**z/VSE can execute in 31-bit mode only. It does not implement z/Architecture, and specifically does not implement 64-bit mode capabilities.**

**z/VSE is designed to exploit select features of IBM eServer zSeries hardware.**

This manual is intended for customers who need to know about the control statements of IBM® z/VSE. It contains a complete description of all IPL, job control, librarian, linkage editor, and MSHP statements and commands.

---

## Who Should Use This Book

This book is mainly intended as a reference source for programmers writing z/VSE™ job control statements and commands.

---

## How to Use This Book

The manual consists of the following sections:

- Chapter 1, "Introduction," on page 1 contains a short description of the different parts of z/VSE and also describes the control statement conventions.
- Chapter 2, "Initial Program Load," on page 7 and Chapter 3, "Job Control and Attention Routine," on page 45 are of interest to anyone using the system, including system analysts, programmers, and operators. Detailed attention routine, job control statement, and job control command formats are given.
- Chapter 4, "Linkage Editor," on page 267 and Chapter 5, "Librarian," on page 285 are of interest to persons responsible for maintaining the resident system. These sections fully describe the control statements for the linkage editor and librarian programs.
- Chapter 6, "Edited Macro Service Program (ESERV)," on page 345 is of interest to programmers using the Assembler language. It describes the control statements necessary to de-edit and update edited macros.
- Chapter 7, "System Buffer Load (SYSBUFLD)," on page 355 is of interest to users who have an IBM 1403U or PRT1 printer attached to their system. The section describes the purpose of SYSBUFLD and how to use it.
- Chapter 8, "Maintain System History Program (MSHP)," on page 365 contains all the MSHP control statements needed for installing and servicing a product. It also describes the control statements intended for IBM personnel (product owners) when preparing a programming package for shipment.
- The Appendix contains a summary of the linkage editor control statements.
- The manual also contains a glossary, and an index.

---

## Where to Find More Information

The control statements for the z/VSE utility programs are not described in this manual. You find these, together with examples, in *z/VSE System Utilities*, SC33-8234, under Control Statement Input.

### **z/VSE Home Page**

z/VSE has a home page on the World Wide Web, which offers up-to-date information about VSE-related products and services, new z/VSE functions, and other items of interest to VSE users.

You can find the z/VSE home page at:

<http://www.ibm.com/servers/eserver/zseries/zvse/>



---

## Summary of Changes

This manual has been updated to reflect enhancements and changes that are implemented with z/VSE Version 3 Release 1. It also includes terminology, maintenance, and editorial changes.

- The name VSE/ESA has changed to z/VSE. However, the names of many features and programs related to z/VSE remain unchanged (such as, for example, IBM COBOL for VSE/ESA, or TCP/IP for VSE/ESA).
- Initial Program Load enhancements:
  - The SCSI IPL and DEF SCSI commands are introduced, to support SCSI disk devices.
  - The IPL ADD command has been enhanced to add HyperSockets devices.
  - The SYS ATL statement has been enhanced with a new TLS parameter.

Job Control enhancements:

- The commands SYSDEF SCSI and QUERY SCSI have been added to support SCSI disk devices.
- The IXFP command has been added for the FlashCopy support that provides high-speed copying support for the IBM TotalStorage Enterprise Storage Server (ESS).
- The LIBSERV command has been enhanced to support the IBM 3494 natively.



---

## Chapter 1. Introduction

This manual contains descriptions of IBM z/VSE system control statements and commands. These statements and commands are grouped by function as shown below.

---

### Initial Program Load

Before a job can be entered into the system for execution, the supervisor and the job control program must be loaded into storage. To do this, the operator starts the system by following the initial program load (IPL) procedure.

---

### Job Control

After the system has been successfully started by means of the IPL procedure, it is ready to accept input for execution. Job control statements are entered on SYSRDR, job control commands at SYSLOG.

The job control program runs in virtual mode in any partition. It is active only between jobs and job steps, and is not present in the partition while a program is being executed.

---

### Attention Routine

When IPL is complete, and the system is running, the attention routine is available at all times. It allows the operator to alter certain system values, query the status of the system, and influence the execution of jobs in the system.

The functions of the attention routine are requested by entering attention routine commands at the console. Some attention routine commands are identical to job control commands, and both types of command are described in alphabetical order in Chapter 3, "Job Control and Attention Routine."

---

### Linkage Editor

Before execution in storage, all programs must be placed in a sublibrary by the linkage editor. An exception to this rule is a single-phase program link-edited with the OPTION LINK. This is linked in VIO space and loaded from there into the partition.

The linkage editor prepares a program for execution by editing the output of a language translator into phase format. The linkage editor also combines separately assembled or compiled program sections or subprograms into phases.

---

### Librarian

The librarian program is used to maintain data which must be readily available to the system, such as programs and cataloged procedures. This data is organized in libraries, which are subdivided into sublibraries, which in turn contain the data, organized in units called members. Each member is identified unambiguously by a library name, sublibrary name, member name and member type. Library,

## Introduction

sublibrary and member names may be freely chosen. The member type depends on the type of data contained in the member.

---

## Edited Macro Service Program (ESERV)

When an assembler macro has been processed (edited) by the assembler, it can no longer be updated directly. Any alterations must be made in the source. If the source of a macro is no longer available, the edited macro must be de-edited. The ESERV program gives you the opportunity to de-edit macros.

---

## System Buffer Load (SYSBUFLD)

SYSBUFLD is a service program for users with IBM 1403U, 3203, 5203, and PRT1 printers. It can be executed as a job or job step to load the Forms Control Buffer (FCB) and/or the Universal Character Set Buffer (UCB) of these printers.

---

## Maintain System History Program (MSHP)

MSHP is a service program needed for installing and servicing an IBM product.

---

## Understanding Syntax Diagrams

This section describes how to read the syntax diagrams in this manual.

To read a syntax diagram follow the path of the line. Read from left to right and top to bottom.

- The ►— symbol indicates the beginning of a syntax diagram.
- The —→ symbol, at the end of a line, indicates that the syntax diagram continues on the next line.
- The ►— symbol, at the beginning of a line, indicates that a syntax diagram continues from the previous line.
- The —→◄ symbol indicates the end of a syntax diagram.

Syntax items (for example, a keyword or variable) may be:

- Directly on the line (required)
- Above the line (default)
- Below the line (optional)

### Uppercase Letters

Uppercase letters denote the shortest possible abbreviation. If an item appears entirely in uppercase letters, it can not be abbreviated.

You can type the item in uppercase letters, lowercase letters, or any combination. For example:

►—KEYW0rd—→◄

In this example, you can enter KEYWO, KEYWOR, or KEYWORD in any combination of uppercase and lowercase letters.

### Symbols

You **must** code these symbols exactly as they appear in the syntax diagram

- \* Asterisk
- :
- Colon

,	Comma
=	Equal Sign
-	Hyphen
//	Double slash
()	Parenthesis
.	Period
+	Add

For example:

\* \$\$ LST

### Variables

Highlighted lowercase letters denote variable information that you must substitute with specific information. For example:

▶▶ `[,USER=user_id]` ▶▶

Here you must code USER= as shown and supply an ID for user\_id. You may, of course, enter USER in lowercase, but you must not change it otherwise.

### Repetition

An arrow returning to the left means that the item can be repeated.

▶▶ `repeat` ▶▶

A character within the arrow means you must separate repeated items with that character.

▶▶ `repeat` ▶▶

A footnote (1) by the arrow references a limit that tells how many times the item can be repeated.

▶▶ `(1) repeat` ▶▶

### Notes:

- 1 Specify *repeat* up to 5 times.

### Defaults

Defaults are above the line. The system uses the default unless you override it. You can override the default by coding an option from the stack below the line. For example:

## Introduction



In this example, A is the default. You can override A by choosing B or C.

### Required Choices

When two or more items are in a stack and one of them is on the line, you **must** specify one item. For example:



Here you must enter either A or B or C.

### Optional Choice

When an item is below the line, the item is optional. Only one item **may** be chosen. For example:



Here you may enter either A or B or C, or you may omit the field.

### Required Blank Space

A required blank space is indicated as such in the notation. For example:

```
* $$ E0J
```

This indicates that at least one blank is required before and after the characters \$\$.

## Frequent Abbreviations

1. `cuu` is the three-digit **device number**, which can be any value between 000 and FFF. It is the number by which the device was defined during I/O configuration.
2. `volser` represents the six-character identifier (the volume serial number) of a tape or disk volume. If you specify less than six characters, the value passed to the system is padded to the left with zeros, unless you enclose the specification in quotes. In this case, the value is padded to the right with blanks. For example,

The specification                      is passed to the system as

<code>VOL1</code>	<code>00VOL1</code>
<code>'VOL1'</code>	<code>VOL1__</code>

Bear in mind that these two specifications will not match when compared by label checking routines. The IPL program always pads to the right with blanks.

For this manual, alphameric characters are defined to include the following: A - Z, 0 - 9, @, \$, and #.

In case of any difference between the conventions given in this manual for control program functions and those appearing in IBM-supplied VSE component publications, observe the deviations given in the component publication.

## Continuation of Commands and Statements

When job control statements are entered through SYSRDR, job control will accept continuation cards or lines only for the ASSGN, DLBL, EXEC, IF, LIBDEF, LIBDROP, LIBLIST, LIBSERV, PROC, PRTY, SETPARM, SETPRT, TLBL, and VTAPE statements. In these statements, up to nine continuation lines are accepted. The operands of the line to be continued may be:

- Entered up to and including column 71, or
- Interrupted after the comma or equals sign separating two operands. Any columns between the interruption and column 72 must contain blanks.

A character string enclosed in single quotes ' ' is regarded as a single operand. Do not interrupt it before column 71, even if it contains commas or equal signs. Position 72 of the line to be continued must always contain a non-blank continuation character (usually a C). The continuation line must start in column 16. For example:

1	16	72
// LIBDEF PHASE,SEARCH=MYLIB.MYSUBA,		C
CATALOG=YOURLIB.YSUBA,		C
TEMP		

When entered through SYSLOG, all job control statements and commands (except those which have no separating commas) and all attention routine commands can be continued on subsequent lines. The existence of a continuation line is indicated by a minus sign immediately following the last delimiting comma on the current line. The command or statement is then continued at the start of the next line.

Example:

```
ALLOC R,F1=128K,-
F2=228K,F3=128K,-
F4=128K
```

Continuation lines may also be entered on SYSLOG in the same way as on SYSRDR.





---

## Chapter 2. Initial Program Load

Operation of the system is initiated through an initial program load (IPL) procedure from the resident disk pack.

When the system enters the wait state, the operator must specify the device to be used for SYSLOG (by pressing END/ENTER), and type in the name of the supervisor to be loaded, together with other information. The format of this information is described in "The Supervisor Parameters Command" on page 11.

The IPL program reads the supervisor into low storage. If a read error occurs while the supervisor is being read, the system goes into a wait state and an error code is set in the first word of processor storage. The IPL procedure must then be restarted.

After successfully reading in the supervisor, the system enters the wait state a second time. The operator then causes an interrupt which, in turn, causes IPL to read its commands from the IPL communication device.

The IPL procedure can be automated almost completely by making use of the Automated System Initialization (ASI) facility. This facility allows all control statements and commands needed for the complete operating system startup to be read from a sublibrary. For guidance on how to code and catalog an ASI procedure, refer to "ASI Procedures" in the *z/VSE Guide to System Functions*. For information on how to execute an ASI procedure, refer to *z/VSE Operation* under "Starting Up the System".

The following section describes the Supervisor Parameters command, followed by the IPL commands in alphabetical order:

- ADD -**  
to define I/O devices
- DEF -** to assign system logical units
- DEF SCSI -**  
to define a SCSI device connection
- DEL -** to delete I/O devices
- DEV -** to display all I/O devices
- DLA -** to define a label information area
- DLF -** to define the lock file
- DPD -** to define the page data set
- SET -** to set the system date and time
- SET XPCC -**  
to enable sharing of DB2<sup>®</sup> data bases between z/VSE and VM/CMS
- SVA -** to increase the SVA size
- SYS -** to set various system options such as number of partitions or channel queue entries.

ADD and DEL must precede any other IPL command, except the SET and SYS commands. DLF (if specified) must be the first command after ADD and DEL. SVA must be the last IPL command.

If the SYS command indicates unattended node support, it must precede the DLA and DPD commands and, depending on other operands, follow the DEF command.

## The IPL Load Parameter

In support of the integrated console the IPL load parameter may be used to specify the preferred system console device, IPL message suppression, IPL prompting, and prompting for the system startup mode.

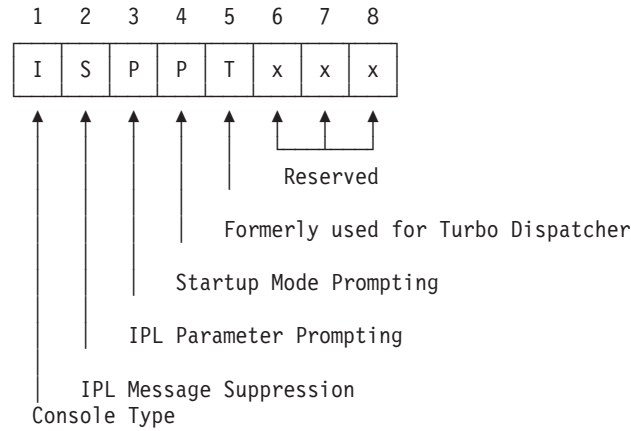


Figure 1. The IPL Load Parameter Format

The **console type** specifies whether the messages are to be routed to an integrated console or to a local console. Possible values for console type are:

### blank

1. If the console specified in the IPL ASI procedure is operational, the messages are routed to that local console.
2. If the local console is not known or not available, the system will wait for an interrupt from a local console.

This is the default.

### .' (Period)

Same as blank.

- L** Route messages to a local console. The console selection is the same as for blank.
- I** Route messages to the integrated console. If the integrated console is not available, then the system selects a local console.
  1. First the system will try to route messages to the integrated console.
  2. In case the integrated console is not available, the system will route messages to the local console specified in the IPL ASI procedure.
  3. If that device is not available, the system will wait for an interrupt from a local console.

The **IPL message suppression code** may be used to request the suppression of messages and command logging during IPL.

You should use it only when initiating a production environment, because you do not necessarily obtain enough information in case of an error. With message suppression you will get the error message only, but preceding messages may be needed to understand the situation. In such a situation it is advisable to repeat the

## IPL Load Parameter

IPL procedure without message suppression. A request for message suppression is ignored during initial installation. Possible values for the IPL message suppression code are:

### **blank**

Display all IPL messages. Print IPL commands on the system console unless the NOLOG option is specified in the supervisor parameters command. This is the default.

### **.' (Period)**

Same as blank.

**S** Suppress all informational messages during IPL. Print only error messages that require a response or an action. Do not print IPL commands on the system console.

The **IPL prompting code** may be used to request a prompting for IPL parameters. It is ignored in a stand-alone environment. Possible values for the IPL prompting code are:

### **blank**

Do not prompt for IPL parameters (the default).

### **.' (Period)**

Same as blank.

**P** Print message 0I03D that prompts for IPL parameters.

The **Startup prompting** code may be used to request a prompting for the system startup mode. If you do not want the system to automatically start the partitions, then you may ask for startup prompting. You will then receive the messages IESI0214I and IESI0215A. As a response to these messages you may select the desired startup mode.

Possible values for startup prompting are:

### **blank**

Do not prompt for the system startup mode. This is the default.

### **.' (Period)**

Same as blank.

**P** Print messages IESI0214I and IESI0215A that prompt for the system startup mode.

The load parameter is a left-justified entry. To decrease the chance of error, a period is accepted as place-holder character. The default values will be chosen for positions that contain a period.

When VSE runs as a VM guest, then specify the IPL load parameter with dots at empty positions, for example:

```
I cuu LOADPARM I.P
```

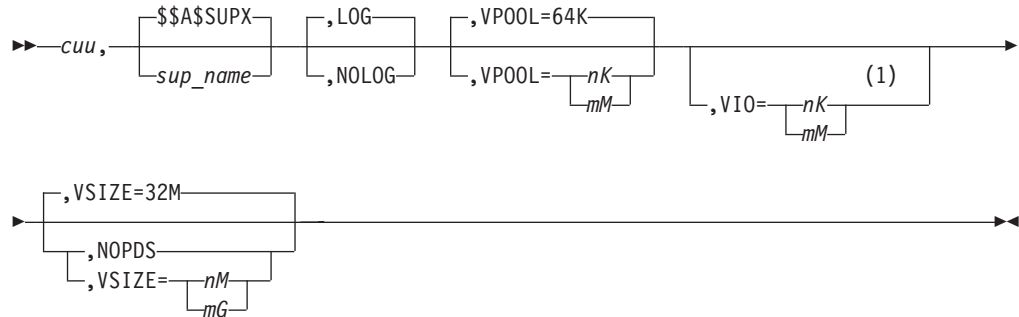
Prior to VSE/ESA 2.3 the **T** parameter was used to activate the **Turbo Dispatcher**. It is no longer used as the Turbo Dispatcher is permanently active.

## The Supervisor Parameters Command

This is the first command to be entered at the console during an interactive IPL, or the first command in the ASI IPL procedure. At the console, it must be entered in response to the message:

```
0I03D ENTER SUPERVISOR PARAMETERS [OR ASI PARAMETERS]
```

Because the system cannot accept any other command at this time, or as the first command in the procedure, the supervisor parameter command has no operation field, just operands. These are as follows:



### Notes:

- 1 The default is the current VPOOL size (rounded to the next higher multiple of 64K).

### cuu

This operand is valid **only** in an ASI IPL procedure. It specifies the physical address of the console device to be assigned to SYSLOG. (During interactive IPL, SYSLOG is defined by pressing END or ENTER on the appropriate device.)

If the system console is an integrated console, **cuu** specifies the device identification to be used for the integrated console. No device may be defined with this number in the IOCDS; if this device number exists, the operand will be ignored.

### sup-name

Specifies the name of the supervisor to be loaded.

### LOG | NOLOG

Controls the logging of IPL commands at the console. If you specify LOG, or omit the operand, the system writes all IPL commands to SYSLOG. If you specify NOLOG, only invalid commands are listed.

### VPOOL=nK | mM

Specifies the size of the V-pool, which is a work area within the SVA needed to exchange data with the virtual I/O area (VIO). The size can be specified in Kilobytes (K) or Megabytes (M); **n** and **m** must be decimal integers. If nK is specified, the system rounds **n** to the next higher multiple of 64K, which is also the default value.

Maximum: 16M (For the actual possible value, see "Storage Allocation Rules" on page 38.)

Minimum: 0K; using the VSE/POWER queue file in VIO, the minimum is 64K.

Default: 64K

## Supervisor Parameters

### VIO=nK | mM

VIO specifies the size of the virtual I/O area, which is a system work area that is part of the page data set (VIO + VSIZE = page data set). The VIO size can be specified in Kilobytes (K) or Megabytes (M); **n** and **m** must be decimal integers, and they must be greater than or equal to the corresponding value in the VPOOL operand. If nK is specified, the system rounds **n** to the next higher multiple of 64.

Maximum: 128M

Minimum: Current VPOOL size (rounded to the next higher multiple of 64K).

Default: Same as minimum.

### NOPDS

Specifies that the system is to operate without a page data set; this can be useful if, for example, enough processor storage is available, or this is simulated by VM. If the operand is omitted, the system requires a page data set. NOPDS must not be specified together with VSIZE.

If a system operates without page data set, it will calculate VSIZE from the size of its processor storage and the requested VIO space, after subtracting approximately 3% for storage management control tables.

### VSIZE=nM | mG

VSIZE specifies the maximum total virtual storage size of a z/VSE system. This includes

- The total size of all shared areas (supervisor, SVA).
- The maximum total size of all static and dynamic partitions which can be allocated concurrently.
- The size reserved for data spaces (including virtual disks). See Note.
- Space for page management requirements: Approximately 4K per 1M VSIZE, rounded up to multiples of segment size. For example, if you specify VSIZE=1G, then about 4M are needed by page management.

**Restriction:** VSIZE must not be specified together with NOPDS.

**Note:** Although initially the total VSIZE is available for address spaces, you must take into consideration that the size reserved for data spaces is taken from VSIZE, too. Thus, VSIZE must be large enough for the virtual address spaces **and** data spaces.

The following VSIZE specifications are possible:

Maximum: 90G

Minimum: 32M

Default: 32M

The largest number accepted for VSIZE is 90G. Although the theoretical maximum value of VSIZE is 90G, the actual maximum VSIZE that can be supported by an installation depends on the device capacity of the device that holds the page data set. Up to 15 extents may be specified for the page data set, and one complete disk may be used as one extent. This means that, for example, 15 IBM 3390 Model 3 disk devices allow a theoretical maximum VSIZE value of 36G.

## ADD

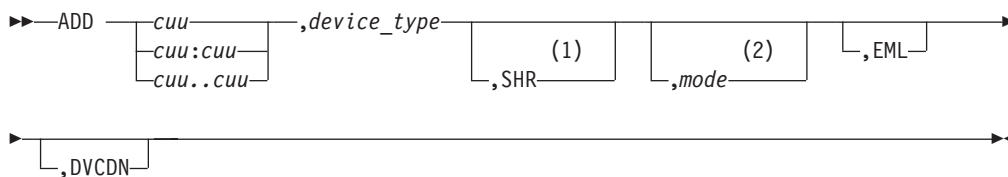
The ADD command is used to define the physical I/O devices attached to the system. The device addresses are entered into the PUB table. Either a single device or a series of devices of the same type can be added with one command.

The ADD command with the FBAV operand is used to define one or more **virtual disks**. A virtual disk emulates an FBA disk. Up to 128 virtual disks can be defined. The layout of the defined virtual disk must be specified with the VDISK command.

The ADD command with the cuu,CONS operands is used to define a non-existing device that will be used internally for the integrated console in case the real system console is to be run in disconnected mode. The command is rejected if the addressed device is defined in the IOCDs.

The ADD command with the cuu,OSAX operand is used to add an Open Systems Adapter (OSA) Express Adapter.

### Format 1



#### Notes:

- 1 SHR can be specified for disk devices only.
- 2 Specification of 'mode' is required for device type 3745.

#### cuu

Indicates the **device number** of the device to be added, which can be any value between 000 and FFF. This is the number by which the device was defined during I/O configuration.

The format **cuu:cuu** or **cuu..cuu** indicates that a series of devices of the same type, starting with the first **cuu** and ending with the second **cuu** is to be added. For example,

```
ADD 130:137,3380
```

defines eight 3380 devices with addresses 130 through 137.

#### device-type

Specifies the device-type code of the device to be defined; see device type codes in Table 1 on page 40.

**Note:** If you specify device type **CTCA** for a channel-to-channel adapter:

- For **cuu**, specify the address of the line attached to the adapter;
- Do not specify the optional operands **mode** or **SHR**.

#### mode

This specification has different meanings for different device types, as follows:

##### For tape devices:

**mode** specifies the mode setting (see Table 6 on page 66). If it is omitted, the following values are assigned:

## ADD

00 for the 3480 and 3490 Magnetic Tape Units  
05 for 3592 Magnetic Tape Units  
08 for the 3490E and 3590 tape units  
90 for 7-track tapes (3420)  
D0 for 9-track tapes (3420)

### For terminal printers:

3284, 3286, 3287, 3288, 3289

**mode** must be entered as 01 (see also Table 1 on page 40).

3745

**mode** must be entered as 01 and specifies a type 5/6 channel adapter.

### SHR

Indicates that the device to be added may be shared by two or more VSE systems. SHR can be specified for disk devices only.

For performance reasons,

- Use this operand only if required
- Put non-shared files on non-shared DASD devices.

### EML

Indicates that the device type as specified by the user should be used to assign the VSE device type code. The EML operand causes IPL to ignore device type sensing, and add the device as the type specified in the ADD command.

### DVCDN

Informs the system that the device to be added is not available for system operation. The operand must be specified in a shared environment in case volume labels are not unique. The purpose is to prevent the system from accessing the wrong device when addressed by volume label.

Do not specify this operand for your SYSRES device, or for the devices containing the lock communications file, the label area or the page data set, or for the primary or alternate IPL device at an unattended node.

## Format 2

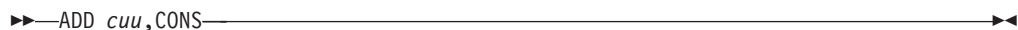


### cuu[:cuu | ..cuu],FBAB

Indicates the device number of a device that is to be used as a **virtual disk**.

The virtual disk is regarded as an FBA device. Any unused cuu of your system can be used, provided it is not overwritten by IPL device sensing.

## Format 3



### cuu,CONS

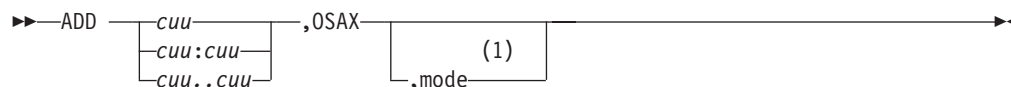
If the system console is *not* an integrated console, this operand specifies the address of a non-existing device (disconnected console) that will be used internally in case the real operator console is to be run in disconnected mode. The command is rejected if the addressed device has been defined in the IOCDs.



If the system console is an integrated console, this operand informs the system that the specified **cuu** is to be used as device identification for the integrated console. Only *one* device of the type CONS may be added. This device number must not be defined in the IOCDS.

If a different console was specified in the IPL ASI procedure, the ADD command specification will override that of the supervisor parameters command. If a device with this number exists, the command will be rejected and the system asks for a device specification not defined in the IOCDS. IPL will not continue processing before a dummy console has been added.

## Format 4



### Notes:

- 1 Specification of mode is required for HiperSockets devices.

### **cuu,OSAX**

Each OSA Express Adapter has to be defined to z/VSE by this command. Specify three device numbers *cuu* with the same number for *c*. Two of the three numbers must follow each other and can be even or odd. For example:

```
ADD D00:D02,OSAX
```

or

```
ADD D00,OSAX
ADD D01,OSAX
ADD D05,OSAX
```

### **cuu,OSAX,1**

To use a HiperSockets connection, three HiperSockets devices are required (one read, one write, and one data device). The corresponding device type is OSAX, but to distinguish HiperSockets devices from OSA Express devices, a mode of 1 must be specified as shown in the example below:

```
ADD C00:C03,OSAX,1
```

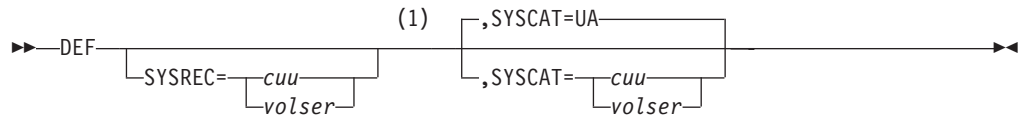
or

```
ADD C03,OSAX,1
ADD C09,OSAX,1
ADD C0F,OSAX,1
```

## DEF

The DEF command, which is mandatory, is used to assign a physical device to

- SYSREC, the logical device for the system recorder file, the hard-copy file, and the job manager file
- SYSCAT, the logical device for the VSE/VSAM master catalog.

**Notes:**

- 1 DEF SYSREC must be specified during IPL.

**SYSREC=cuu | volser**

Indicates the device number or the volume serial number of the system recorder file. Please note that the STDLABELs in the label area on the volume (see DLA command on page 20) must match the DEF SYSREC=... definition.

**SYSCAT=cuu | volser | UA**

Indicates the device number or the volume serial number of the VSE/VSAM master catalog.

UA indicates that the logical device is to be unassigned. This is the default value.

The assignments cannot be changed until the next IPL.

## DEF SCSI

The DEF SCSI command is used to associate the VSE SCSI device number (FBA) with the real SCSI Logical Unit Number (LUN), and its connection path (FCP, WWPN).

For each SCSI device a DEF SCSI command has to be included. In case the same SCSI device is attached via additional FCP devices (multipathing), a separate DEF SCSI command is required for each path.

Each DEF SCSI command causes the system to connect to the specified SCSI device. If the connection cannot be established because of an incorrectly specified configuration, the command can be reentered with corrected configuration parameters.

It is advisable to place the DEF SCSI commands right after the ADD/DEL commands, because the DEF SCSI commands must be given before the first access to a SCSI device. Therefore, when the page data set, or any of its extents, or the label area, or the hardcopy file, or the VSAM master catalog are allocated on a SCSI disk, the DEF SCSI command must precede the DPD, DLA, DEF SYSREC, or DEF SYSCAT commands.

►►—DEF SCSI,—FBA=*cuu*,—FCP=*cuu*,—WWPN=*portname*,—LUN=*lun*—◄◄

### **FBA=*cuu***

*cuu* is the SCSI device ADDED as FBA.

### **FCP=*cuu***

*cuu* is the device number of the attaching FCP ADDED as FCP.

### **WWPN=*portname***

*portname* is the 64 bit world wide port name of the SCSI controller configured to access the LUN.

It is specified in 16 hexadecimal digits. Valid specifications are 0 to 9 and A to F.

### **LUN=*lun***

*lun* is the 64 bit logical unit number identifying the particular SCSI device as configured in the SCSI controller.

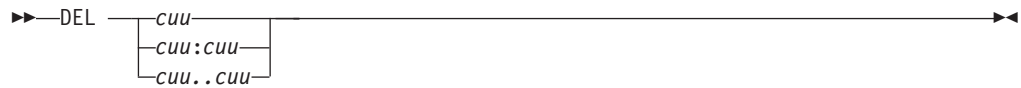
It is specified in 1 to 16 hexadecimal digits. Valid specifications are 0 to 9 and A to F. If less digits than 16 are specified, trailing zeros will be presumed. For example, LUN 216B0000 00000000 can be specified as LUN=216B.

## DEL

---

## DEL

The DEL command is used to delete one or more of the I/O devices previously defined with the ADD command.



### **cuu**

Indicates the device number of the device(s) to be deleted.

The format **cuu:cuu** or **cuu..cuu** indicates that a series of devices of the same type, starting with the first **cuu** and ending with the second **cuu** is to be deleted. For example,

```
DEL 130:133
```

causes devices 130, 131, 132, and 133 to be deleted.

---

## DEV

The DEV command is used to display all I/O devices sensed and added. This allows to identify all devices which are not needed.

This is especially useful if more than 1024 devices are operational during IPL. In this case the system issues message 0J74D which through an internal DEV command shows the complete I/O configuration. Devices not needed can then be deleted by DEL commands.

The device type is either shown as specified by the ADD command or as the control unit of the device returns it. If a device type cannot be determined it is shown as *UNSP*.

▶—DEV —————▶

In the following clip of a DEV output example, devices 009, 00C to 00E, 150 and 151 show the device type specified on the ADD command.

```
BG 0000 DEVICES ADDED AND/OR SENSED:
BG 0000 CUU RANGE  DEVICE TYPE
BG 0000 009        3277
BG 0000 00C        2540R
BG 0000 00D        2540P
BG 0000 00E        1403
BG 0000 150:151    ECKD
```

The DEV command may be given as long as ADD and DEL commands are accepted, that is until the IPL restart point is bypassed.

## DLA

The DLA command, which is mandatory, defines or references a label information area. This area may be located on any physical disk device. Format and layout of the label information area are determined by the system.

The DLA command must be entered after the ADD (and DEL) commands and before the SVA command. Only one valid DLA command may be entered.

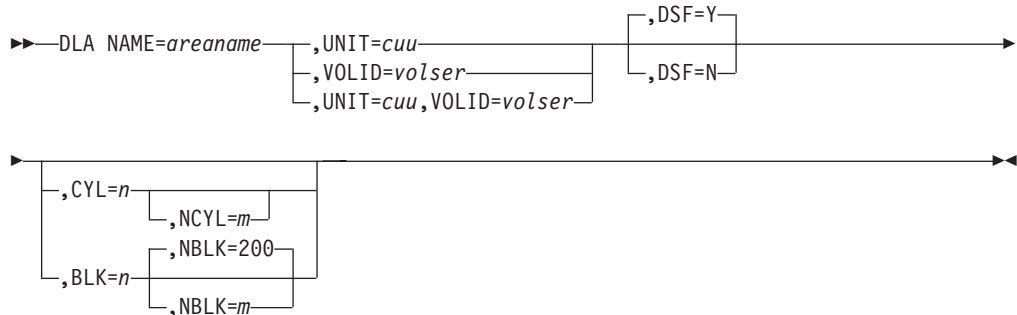
After completion of the IPL procedure the VDISK command with the USAGE=DLA parameter has label information stored in native dataspace. See the *z/VSE Planning* manual for further information.

If you want to use previously created standard labels, enter either:

- A short form of the DLA command with only the NAME= and UNIT= or VOLID= operands, or
- A long form of the DLA command with exactly the same operands as the command used to create the standard label area.

To clear the standard label area, enter the long form of the DLA command with the same CYL and NCYL (or BLK and NBLK) operands, but a different NAME operand. The system issues the message **OVERLAP ON filename**. Reply DELETE to format the new label area.

If you specify CYL and NCYL (or BLK and NBLK) different from those of an existing label area, the system issues the message **DUPLICATE NAME ON VOLUME**. If you want to use the new extent, reply DELETE to this message.

**NAME=areaname**

Specifies the name of the label area, which can be one to eight alphameric characters. When the label area is first created, this name is entered in the VTOC of the device indicated in the UNIT or VOLID operand (in the form: **DOS.LABEL.FILE.cpu-id.areaname**). When referring to the label area during subsequent IPLs, use only the NAME and UNIT or VOLID operands.

**UNIT=cuu**

Specifies the device number of the device containing the label area. The device type may be different from that of the SYSRES device. This operand may be specified together with VOLID.

**VOLID=volser**

Identifies the unique volume serial number of the device containing the label area. This operand may be specified together with UNIT.

**DSF=Y | N**

Specifies whether the label area is to be data-secured. If the operand is omitted, DSF=Y (Yes) is assumed.

**CYL=*n***

Indicates, for CKD devices, the sequential number of the cylinder, relative to zero, where the label area is to begin. *n* must be a decimal number with one to five digits, with a minimum value of 1 and a maximum value of 32767.

**NCYL=*m***

Defines, for CKD devices, the size of the label area in number of cylinders. *m* must be a decimal number with one to five digits, with a minimum value of 1 and a maximum value of 32767. If this operand is omitted, a default size of 3 is used. The maximum value for *m* is 16 for all other devices.

**BLK=*n***

Indicates, for FBA devices, the sequential number of the block, relative to zero, where the label area is to begin. *n* must be a decimal number with one to 8 digits, with a minimum of 2, but see the note on the NBLK operand below.

**NBLK=*m***

Defines, for FBA devices, the size of the label area in number of blocks. *m* must be a decimal number with a minimum of 12 and a maximum of 992.

**Note:** The sum of the NBLK specification and the BLK specification must be less than 558 000.

If this operand is omitted, the default, which is 200 blocks, is used.

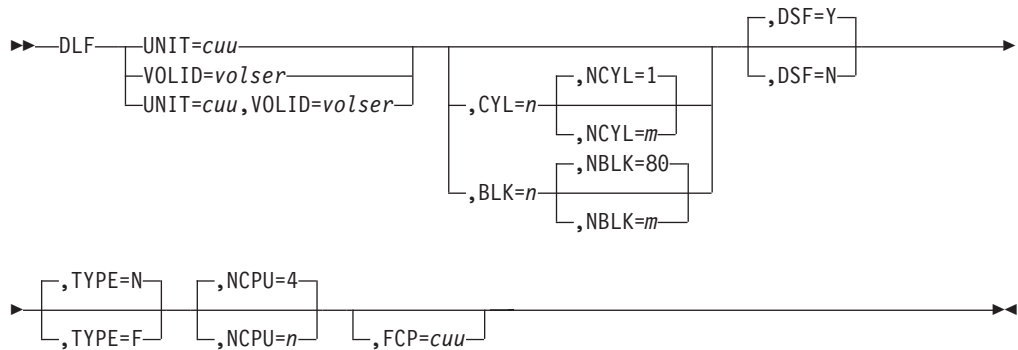
## DLF

The DLF command is used to define or reference the cross-system communication file (lock file). This file must exist when two or more VSE systems share disk storage devices. The DLF command is required if devices which are defined with the SHR option in the ADD command exist.

If the lock file has to be allocated on a SCSI disk, the following hardware restriction applies: CPUs sharing the lock file must not access it via the same physical FCP adapter. See *z/VSE Planning* for more information.

Do not allocate the lock file on a SCSI-FBA DOSRES or SYSWK1 disk. If you choose one of these volumes, the DLF command will be rejected.

The lock file has to be on a disk drive which is physically shared with all systems linked in the disk sharing environment. If used, the DLF command must be the first command after the ADD (and DEL) commands, or - in case SCSI devices exist - after the DEF SCSI commands.

**UNIT=cuu**

Specifies the device number of the device containing the lock file. This operand can be used together with VOLID.

**VOLID=volser**

Identifies the unique volume serial number of the disk containing the lock file. This operand can be used together with UNIT.

No operands other than UNIT or VOLID are needed if an existing lock file is to be used. If, however, a new lock file is to be created or if a reallocation is required, the following operands are also needed:

**CYL=n**

Specifies, for CKD devices, the sequential number of the cylinder, relative to zero, where the lock file is to begin. **n** must be a decimal number with one to five digits, with a minimum value of 1 and a maximum value of 32767.

**NCYL=m**

Specifies how many cylinders of a CKD device are to be allocated to the lock file. **m** must be a decimal number with one to five digits, with a minimum value of 1 and a maximum value of 32767. The default is also 1. For details, see the formula shown at the end of this section.

**BLK=n**

Specifies, for FBA devices, the sequential number of the block, relative to zero, where the lock file is to begin. **n** must be a decimal number with 1 to 8 digits, with a minimum of 2.



**NBLK=m**

Specifies how many blocks of an FBA device are to be allocated to the lock file. The default is 80. For details, see the formula shown at the end of this section. **m** must be a decimal number with 1 to 8 digits.

**DSF=Y | N**

Specifies whether the lock file is to be data-secured. If the operand is omitted, DSF=Y (Yes) is assumed.

**TYPE=N | F**

**N**, which is default, indicates that the lock file is not to be formatted. If you specify TYPE=N, but the lock file does not exist on the specified device or volume, the system ignores the operand and formats a new lock file.

**F** indicates that the system should format the lock file during IPL. Use this option only when a new lock file must be formatted, for example, because of an error in the existing one. Be sure to enter a DLF command with the TYPE=F operand at only one of the CPUs sharing the lock file.

**NCPU=n**

Specifies the number of machines, real or virtual, which share disk storage. Valid specifications for **n** are 2..31. The default is 4.

**FCP=cuu**

*cuu* is the device number of the FCP adapter that connects to the SCSI disk containing the lock file. It must be specified to confirm the correct installation and configuration of the connection.

If you want to allocate a lock file on a SCSI disk, you must have a unique FCP adapter installed for each CPU sharing the lock file, and access the lock file via this unique FCP. Only one connection path can be defined to access the lock file.

The operand is required for FBA-SCSI and will be ignored for ECKD or any other FBA device.

**Note:** If you want to use a previously created lock file, enter either:

- A DLF command with only the UNIT= and/or VOLID= operands, or
- A DLF command with exactly the same operands as the command used to create the existing lock file.

To reformat the existing lock file, enter the long form of the DLF command with the same CYL and NCYL (or BLK and NBLK) operands, but with the operand TYPE=F.

If you specify CYL and NCYL (or BLK and NBLK) different from those of an existing lock file, the system issues the message **DUPLICATE NAME ON VOLUME**. If you want to use the new lock file, reply DELETE to this message.

The maximum number of resources that can be locked by a lock file of a given size can be calculated by the following formulas:

- For FBA devices:  
Number of resources =  $NBLK \times (508 \div (12 + NCPU))$
- For CKD devices:  
Number of resources =  $NCYL \times [(508 \div (12 + NCPU)) \times D]$

where D is the number of physical blocks per cylinder:

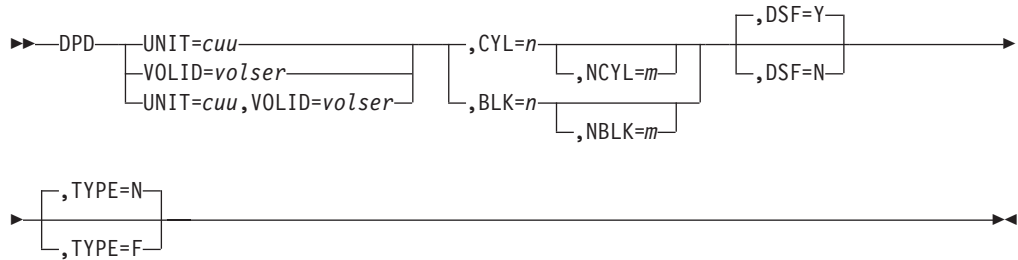
690 for IBM 3380  
720 for IBM 3390

---

**DPD**

The DPD command is used to define the page data set (PDS). The PDS is required to store paged-out pages of programs executing in virtual mode. The size of the PDS depends on the amount of pageable address space.

The command is invalid in an environment without page data set (NOPDS option in supervisor parameters command).



The operands of the DPD command may be given in any order.

**UNIT=cuu**

Specifies the device number of the device that is to contain the page data set. You may specify this operand together with VOLID.

**VOLID=volser**

Identifies the volume serial number (one to six alphabetic or numeric characters) of the disk pack that contains the page data set. If you do not specify VOLID, the volume serial number is not checked. You may specify this operand together with UNIT.

**CYL=n**

Specifies, for CKD devices, the sequential number of the cylinder, relative to zero, where the page data set is to begin (in decimal). A specification of CYL=0 indicates that the page data set extent is to begin on cylinder 0, track 1. **n** must be a decimal number with one to five digits, with a maximum value of 32767.

**NCYL=m**

Specifies, for a multi-extent CKD page data set, the size of one page data set extent (in number of cylinders). **m** must be a decimal number with one to five digits, with a minimum value of 1 and a maximum value of 32767.

**BLK=n**

Specifies, for FBA devices, the sequential number of the block, relative to zero, where the page data set is to begin. **n** must be a decimal number with 1 to 8 digits, with a minimum of 2.

**NBLK=m**

Specifies, for a multi-extent FBA page data set, the size of one page data set extent (in number of blocks). **m** must be a decimal number with a minimum of 64; it should also be specified as a multiple of 64.

**DSF=Y | N**

Indicates whether the page data set is to be data-secured. Yes is the default. For multi-extent page data sets, the DSF specification is valid for the **first** extent definition only; it is ignored for any further extent definitions.

**TYPE=N**

TYPE=N is the default and indicates that the page data set need not be formatted. The TYPE operand is ignored for FBA devices.

If TYPE=N is specified, but the page data set does not exist, or the extent limits have been changed, TYPE=N is ignored and the page data set is formatted during IPL.

**TYPE=F**

Indicates that the page data set is to be formatted during IPL. Formatting during IPL is required if the page data set has been damaged. The TYPE operand is ignored for FBA devices.

For each extent of a multi-extent page data set, a separate DPD command has to be entered. After each command, the operator will be prompted to enter the next extent definition until

- The entire virtual storage is mapped on the specified extents, or
- The maximum number of extents allowed (which is 15) is exceeded, or
- The operator enters a DPD command without the NCYL/NBLK operand, in which case the complete remaining storage will be mapped on this extent.

Up to 15 extents can be specified, and each extent is allocated in multiples of eight 4K-records. The extents may reside on different volumes; up to three extents may be allocated on one volume. The various extents can be placed on different CKD device types, or can be mixed with FBA device extents.

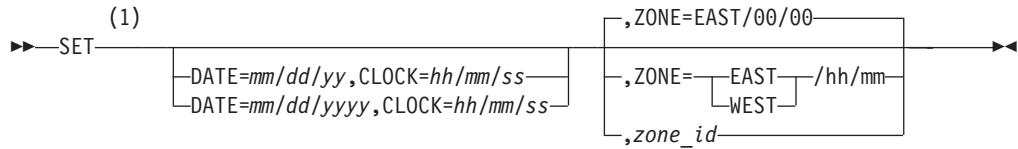
The size of the page data set must be equal to the amount of virtual storage defined in the supervisor parameters VSIZE and VIO. (Remember that VSIZE includes all address spaces **and data spaces**, including virtual disks).

If the size specified in the NCYL/NBLK operand is larger than the size actually needed for the page data set, the free cylinders/blocks are available to the user.

---

**SET**

The SET command, which is optional, is used to set the system date, the time-of-day (TOD) clock, and the system time zone. It is required only if the TOD clock has not been set since the last POWER ON; IPL will then prompt the operator to enter the SET command. The command may be entered at any time before the SVA command.

**Notes:**

- 1 At least one operand must be specified.

**DATE=mm/dd/yyyy**

Specifies the date in months (1-12), day of the month (1-31), and year (4 digits). For compatibility reasons, the specification of only the last two digits of the year is still accepted. In this case, a number above 50 is interpreted as 19yy and a number below or equal 50 as 20yy.

The highest DATE and CLOCK value that can be specified is

```
SET DATE=09/17/2042,CLOCK=23/53/47
```

Any higher value causes a TOD clock overflow, and is rejected.

After IPL this format can be changed to dd/mm/yyyy with the STDOP command.

**CLOCK=hh/mm/ss**

Specifies the local time-of-day in hours, minutes and seconds.

**ZONE=EAST/hh/mm**

Specifies that the installation is located at a geographical position east of Greenwich.

**ZONE=WEST/hh/mm**

Specifies that the installation is located at a geographical position west of Greenwich.

**hh/mm**

Indicates the difference in hours and minutes between local time and Greenwich Mean Time. hh may be in the range 0-23, mm in the range 0-59.

**zone\_id**

Is a three character time zone definition (for example, EST or EDT) established by an earlier SET™ ZONEDEF statement.

The operands that have to be specified with the SET command depend upon the state of the TOD clock. The following groups can be distinguished:

1. If the TOD clock is in the set state, the command may be given in one of the three forms:

```
SET ZONE=
SET DATE= ,CLOCK=
SET DATE= ,CLOCK= ,ZONE=
```

2. If the TOD clock is in the not-set state, the command **must** be given in one of the two forms:

```
SET DATE= ,CLOCK=  
SET DATE= ,CLOCK= ,ZONE=
```

Normally, the TOD clock is up and running and should not be modified by a SET DATE=, CLOCK= or ZONE= command, unless you want to do it for test purposes. Normally, in your IPL procedure, you would only include a SET ZONE= command in order to specify for your system the local time difference to GMT. However, if you want to make use of the standard and daylight saving time feature described below, you must not include this SET command format in your IPL procedure.

**Notes:**

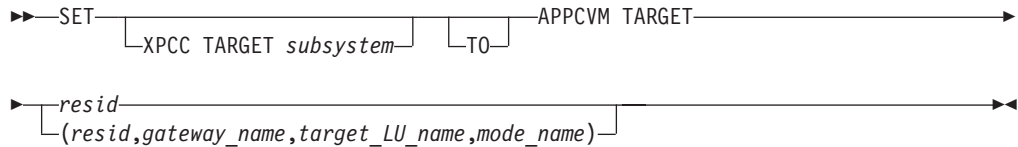
1. If the TOD clock is in the set state, message 0I30I is printed. If the TOD clock is in the not-set state, message 0I31I is printed. If the TOD clock is inoperative, message 0I32A is printed, and IPL terminates.
2. VM users: if you want to change the clock date on a VSE system running under VM, make sure that the directory entry of your virtual machine has TODENABLE set. Otherwise the clock and date changes will be ignored.

The time-of-day clock should always hold the exact time, that is, the time that has elapsed since January 1, 1900, 00.00 hrs.

## SET XPCC

This command is used to activate the VSE XPCC/APPC/VM support, which enables VSE DB2 applications to share one or more DB2 data bases with applications running on other VM guest machines.

A separate SET command must be specified for each VM resource with which VSE wants to establish communication. Up to ten VM resources may be specified.



### subsystem

Specifies the name of the subsystem the VSE application wants to communicate with. If more than one SET command is given, the subsystem name must be unique.

The subsystem name is the same as that specified in the corresponding assembler macro XPCCB TOAPPL=application-name. The name may be up to 8 characters long. The character string SYSARI (and any character string starting with SYSARI) is reserved for DB2 and may not be used for any other subsystem name.

### resid

Is the name of the resource defined in VM as APPC/VM communication partner. For DB2 the resource name is the name of the DB2 data base in VM.

### (resid, gateway-name, target-LU-name, mode-name)

Describes the network routing information if the target VM resource is linked through a VM gateway.

'gateway-name, target-LU-name, mode name' describe the connection to the target VM resource in an SNA network. The names are defined by VTAM\* statements when the network is built.

The following examples illustrate the naming rules:

```

SET XPCC TARGET subsystem1 TO APPCVM TARGET resid1
SET XPCC TARGET subsystem2 TO APPCVM TARGET resid2
SET APPCVM TARGET (resid3,g3,tlu3,mode3)
SET APPCVM TARGET (resid4,g4,tlu4,mode4)
  
```

where

**subsystem1**, **subsystem2**, **resid3**, and **resid4** have to be unique names. **resid1** and **resid2** do not have to be unique names which means that, for example, **resid1** may have the same name as **resid2** and/or **resid3**.

If activation of the VSE XPCC/APPC/VM communication fails, a message is issued and IPL processing continues. All subsequent XPCC requests that try to establish an APPC/VM connection are rejected with an error indication.

## SET ZONEDEF / SET ZONEBDY

The purpose of SET ZONEDEF and SET ZONEBDY is to be able to switch between standard and daylight saving local times without changing the IPL startup procedure each time. Note that you have to IPL the system in order to switch to the new time zone. The *Tailor IPL Procedure* dialog helps you to define these commands.

Switching is achieved by selecting a system zone from a set of zone definitions and zone boundary definitions included in the IPL startup procedure. These definitions are provided through the IPL commands SET ZONEDEF and SET ZONEBDY. The definitions are saved by IPL in a zone boundary definition table, for immediate or later access.

Make sure that the IPL process does not contain a SET command with the operands DATE=, CLOCK= and (or) ZONE= because automatic time zone selection at IPL will not be done, if date, clock or zone are set explicitly.

In case the TOD clock is not operational at IPL, the operator will be prompted to enter the SET DATE=, CLOCK=, ZONE= command.

The commands are to be used as follows:

- SET ZONEDEF

Use the SET ZONEDEF command to define system time zones according to their difference from Greenwich Mean Time (GMT).

- SET ZONEBDY

Use the SET ZONEBDY command to tell z/VSE what time zone to choose at IPL so that it can determine the local time.

Zone and zone boundary definitions are supplied at IPL by the SET ZONEDEF and SET ZONEBDY commands. Zone selection, when applicable, occurs after the latest opportunity to enter a normal SET command has passed. The TOD clock has to be in the set state.

## SET ZONEDEF

```
▶▶ SET ZONEDEF, ZONE= EAST /hh/mm, zone_id ▶▶
    WEST
```

Use the SET ZONEDEF command to define system time zones according to their difference from Greenwich Mean Time (GMT).

Include as many statements as needed up to a maximum of 10. They are optional. Usually, two statements are needed to define standard (winter) time and daylight saving (summer) time. You can place them anywhere in the IPL procedure before the SVA command. However, it is recommended to place them after the ADD/DEL statements, because the zone ID has to be defined before a SET ZONEBDY or a SET DATE= command refers to it.

If you specify more than one statement with the same operands, the last statement overrides any previous specifications.

The following operands can be specified:

## SET ZONEDEF / SET ZONEBDY

### ZONE

EAST tells z/VSE to add the specified hh/mm value to Greenwich Mean Time (GMT) to define this time zone ID.

WEST tells z/VSE to subtract the specified hh/mm value to Greenwich Mean Time (GMT) to define this time zone ID.

hh/mm is the difference between GMT and zone ID in hours and minutes.

### zone\_id

Is a three character string providing a name for a zone definition, like EST and EDT. It can be used in SET commands to refer to a specific zone value.

## SET ZONEBDY

```
▶▶—SET—ZONEBDY—,DATE=mm/dd/yyyy—,CLOCK=hh/mm/ss—,zone_id————▶▶
```

Use the SET ZONEBDY command to tell z/VSE what time zone to choose at IPL so that it can determine the local time.

Include as many statements as needed up to a maximum of 20. They are optional. You can place them anywhere before the SVA command, but they must follow after the SET ZONEDEF statement that establishes the zone ID referred to.

If you specify more than one statement with the same time value, the last statement overrides any previous specification.

The following operands can be specified:

### DATE

Is the date, in the format mm/dd/yyyy, on which z/VSE should begin using a given time zone.

### CLOCK

Is the local time, in the format hh/mm/ss, on which z/VSE should begin using a given time zone.

### zone\_id

Is a three character time zone definition established by an earlier SET ZONEDEF statement.



**Examples**

1. To define central European summer time (CES) and central European standard time (CET) from the year 1997 till the year 2000, add the following statements to your IPL procedure:

```
ADD ... (last ADD command)
SET ZONEDEF,ZONE=EAST/02/00,CES
SET ZONEDEF,ZONE=EAST/01/00,CET
SET ZONEBDY,DATE=03/30/1997,CLOCK=02/00/00,CES
SET ZONEBDY,DATE=10/26/1997,CLOCK=02/00/00,CET
SET ZONEBDY,DATE=03/29/1998,CLOCK=02/00/00,CES
SET ZONEBDY,DATE=10/25/1998,CLOCK=02/00/00,CET
SET ZONEBDY,DATE=03/28/1999,CLOCK=02/00/00,CES
SET ZONEBDY,DATE=10/31/1999,CLOCK=02/00/00,CET
SET ZONEBDY,DATE=03/26/2000,CLOCK=02/00/00,CES
SET ZONEBDY,DATE=10/29/2000,CLOCK=02/00/00,CET
```

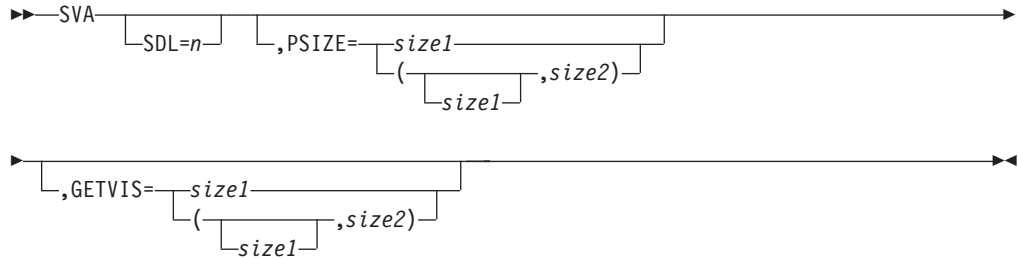
2. To define eastern daylight time (EDT) and eastern standard time (EST) from the year 1997 till the year 2000, add the following statements to your IPL procedure:

```
ADD ... (last ADD command)
SET ZONEDEF,ZONE=WEST/04/00,EDT
SET ZONEDEF,ZONE=WEST/05/00,EST
SET ZONEBDY,DATE=04/06/1997,CLOCK=02/00/00,EDT
SET ZONEBDY,DATE=10/26/1997,CLOCK=02/00/00,EST
SET ZONEBDY,DATE=04/05/1998,CLOCK=02/00/00,EDT
SET ZONEBDY,DATE=10/25/1998,CLOCK=02/00/00,EST
SET ZONEBDY,DATE=04/04/1999,CLOCK=02/00/00,EDT
SET ZONEBDY,DATE=10/31/1999,CLOCK=02/00/00,EST
SET ZONEBDY,DATE=04/02/2000,CLOCK=02/00/00,EDT
SET ZONEBDY,DATE=10/29/2000,CLOCK=02/00/00,EST
```

## SVA

This command is mandatory and must be the last command entered during the IPL procedure. It is used to allocate space within the SVA into which the user can later load his phases. The values specified in the SVA command are added to the system SVA space requirements, which depend on the supervisor being used.

All operands are optional. If the operands are not entered during IPL, there will be no space reserved in the SDL and SVA for user phases. However, an SVA of sufficient size to contain the required set of system phases and the default system GETVIS area will be created.



### SDL=n

Specifies the decimal number of entries in the system directory list to be reserved for user phases and for the SVA-eligible phases of z/VSE components which are **not** loaded automatically at IPL. Do not specify entries for the phases loaded automatically during IPL, as this is done by IPL. This applies both for SVA-24 or SVA-31 phases.

Because of rounding, more SDL entries may be reserved than specified. A message will display the number of SDL entries actually available to you.

The maximum number of SDL entries reserved by the system is 32765. If your SDL=n specification plus the number of SDL entries for the automatically loaded phases exceeds 32765, a warning message is issued displaying the number of SDL entries actually available to you.

The SDL is allocated in the SVA-24. Therefore do not specify a much larger number of SDL entries than is actually needed in order to avoid wasting shared space below 16M. Approximately 56 SDL entries fit in a 4KB block of storage.

Note that at IPL time only phases from IJSYSRS.SYSLIB and those generated with the SVA or SVAPFIX operand in the linkage editor PHASE statement can be loaded into the SVA.

### PSIZE=size1 | ([size1],size2)

Specifies the size of the area within the SVA which is to be reserved for **user** phases and for the SVA-eligible phases of z/VSE components which are **not** loaded automatically at IPL. (Do not specify space for the phases loaded automatically into the SVA during IPL, as IPL will reserve the necessary space.) The specified size should be large enough for the user phases and for a maintenance area which is required when a phase with a copy in the SVA is replaced.

**size1** can be coded either as nK or mM and specifies the user space required in the 24-bit addressability SVA.

**size2** can be coded either as xK or yM and specifies the user space required in the 31-bit addressability SVA.

The 24-bit part of the SVA is allocated adjacent to the supervisor, the 31-bit part - if it exists - is at the high end of the address space. (For details of the storage layout, see “Chapter 1. Storage Management” in the *z/VSE Guide to System Functions*.)

- For a 24-bit SVA the size can be specified either in n Kilobytes (K) or m Megabytes (M).
  - Maximum: 16384K or 16M (For the actual maximum values see “Storage Allocation Rules” on page 38.)
  - Minimum: 0
  - Default: 0
- For a 31-bit SVA the size can be specified either in x Kilobytes (K) or y Megabytes (M).
  - Maximum: 2097152K or 2048M (For the actual maximum values see “Storage Allocation Rules” on page 38.)
  - Minimum: 0
  - Default: 0

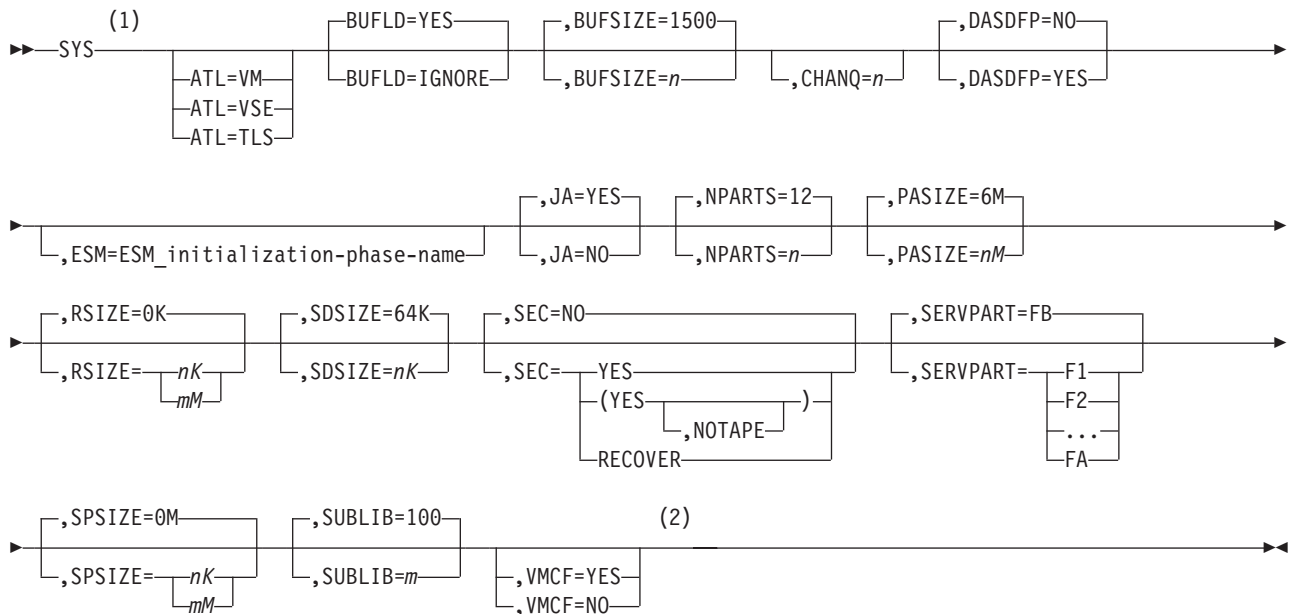
#### **GETVIS=size1 | ([size1],size2)**

Indicates the size of an additional, **user** system GETVIS area which you can specify beyond the minimum size allocated by the system. (The system automatically reserves GETVIS space for its own requirements.)

The size must be specified in the same way as for the PSIZE operand. See also “Storage Allocation Rules” on page 38.

## SYS

The SYS command, which is optional, specifies various system options, such as number of partitions or supervisor buffers.



## Notes:

- 1 At least one operand must be specified. The operands can be specified in different SYS commands.
- 2 The operand is valid only on a VM host.

**ATL=VM | VSE | TLS**

VM indicates that the automatic tape library is supported by VM via the VSE Guest Server (VGS). This is the default when the system is IPLed as a VM guest. The operand is invalid for native VSE.

VSE indicates that the automatic tape library is supported natively by VSE. VSE is the default when the system runs native. This operand is required when the system is IPLed as a VM guest, but the automatic tape library is supported via the Library Control Device Driver (LCDD) on VSE.

TLS indicates that an automatic tape library is supported by the native VSE Tape Library Support. LCDD and VGS are not needed to support the 3494 tape library.

**BUFLD=YES | IGNORE**

Specifies what action the system is to take if a printer which supports automatic print-control-buffer loading is not READY during IPL.

If you specify BUFLD=YES, or omit the operand, IPL stops, issues a console message, and waits for the requested operator action before continuing.

If you specify BUFLD=IGNORE, IPL continues without waiting for operator action.

**BUFSIZE=n**

Specifies the **number** of supervisor buffers to be used for I/O processing (CCW translation).

**n** must be a decimal number with a maximum of seven digits. The minimum value for **n** is 10, the default is 1500.

**Note:** Because of system requirements, the actually allocated number of supervisor buffers may be much larger than the number specified. Before IPL completes, a message displays the actual BUFSIZE value.

#### **CHANQ=n**

Specifies the number of channel queue entries to be allocated. If you omit the operand, the system allocates the appropriate number of channel queue entries for the number of system tasks, and the type and number of devices added, as shown in the following table:

	IODEV<255	IODEV≥255
MINIMUM	$n=(a+s)$ or $255*$	$n=(a+s)$
DEFAULT	$n=(a+5d+s)$ or $255*$	$n=(a+5d+s)$
MAXIMUM	$n=255$	$n=32767$

IODEV is a supervisor generation option that indicates the maximum number of I/O devices to be attached.

\* whichever is smaller

a= total number of ADDED devices  
d= number of disk devices added  
s= number of system tasks

**Note:** Because of system requirements, the actually allocated number of channel queue entries may be larger than the number specified. Before IPL completes, a message displays the actual CHANQ value.

#### **DASDFP=YES | NO**

Specifies whether disk file protection should be active. If you omit the operand, the system does not activate file protection.

Using DASD file protection may only be beneficial for applications that generate their own channel programs.

#### **ESM=ESM\_initialization\_phase\_name**

Specifies the name of the External Security Manager (ESM) initialization phase. If the operand is not specified or the specified phase name is invalid, the Basic Security Manager (BSM) will be activated. See the *z/VSE Planning* manual for detailed information about the BSM and ESM.

If an ESM phase name has been specified in the IPL ASI procedure but you want to start your system with the BSM active, override the SYS ESM= phasename command by SYS ESM= (nothing specified).

#### **JA=YES | NO**

Specifies that CPU-times and I/Os for all devices are accounted. Job accounting is always activated, even if NO is specified. NO is accepted for compatibility reasons.

#### **NPARTS=n**

Specifies the maximum number of partitions that can be activated concurrently. The number includes static and dynamic partitions. **n** is a decimal number between 12 and a system-defined maximum of about 200. The default is 32.

Do not specify a much larger number of partitions than is actually needed in order to avoid wasting resources.

#### **PASIZE=nM**

Specifies the maximum size of the private area within an address space. The private area is the portion of an address space that is available for the allocation of private partitions.

The PASIZE value should correspond to either the size of the largest dynamic or static private partition to be allocated or, if the total size of the partitions to be allocated in the same address space is larger, the sum of these partitions.

The minimum size of the private area depends either on VSIZE or, in an environment without page data set, on the processor storage size. In any case, the private area must be large enough to hold the page manager tables.

Maximum: 2048M

Minimum: 1M if VSIZE is smaller than 256M

Minimum: 6M if VSIZE is 256M or larger

Default: 6M

In an environment without PDS, VSIZE has approximately the value of processor storage minus VIO space. The maximum of 2048M is only a theoretical value, since it does not consider shared areas. (If you specify a value larger than 2048, the minimum value of 6M is assumed.) At least 1M of the private area has to be allocated below 16M. The system terminates if this is not guaranteed.

The private area may become larger than specified because of system requirements. However, the size of the private area and of the shared areas together must not be larger than either 2048M or VSIZE (if VSIZE is smaller than 2048). Otherwise the system will decrease the PASIZE value to the largest possible size and issue a message displaying the actual value of PASIZE.

See also "Storage Allocation Rules" on page 38.

#### **RSIZE=nK | mM**

Specifies the amount of real storage that may be allocated for programs that are to be executed in real mode. This storage has to be available below 16M.

RSIZE is required if you allocate real partitions with the ALLOC R command. The RSIZE value should correspond to the sum of all areas allocated by ALLOC R.

Maximum: 16384K or 16M

Minimum: 0K

Default: 64K

If specified in nK bytes, the value n is rounded to the next higher multiple of 4.

**Note:** RSIZE is required whenever an ALLOC R command is used in the startup job stream. However, ALLOC R should only be used for allocating real partitions that are needed for real execution. If your program needs PFIX storage, replace the ALLOC R command by the SETPFIX command.

#### **SDSIZE=nK**

Specifies the size of a shared V=R area for system monitor functions, for example SDAID. The suggested SDSIZE for SDAID is 64K (default). It will not run in 0K. Valid specifications for n are:

Maximum: 256

Minimum: 0

Default: 64

n is rounded to the next higher multiple of 4.

#### **SEC=YES | NO | (YES[,NOTAPE]) | RECOVER**

Specifies which type of security is to be activated.

If **YES** is specified, the installed security manager will perform access authorization checking for resources as defined in DTSECTAB like files or libraries. In addition, access logging is activated, if BSM is the security manager and if the z/VSE optional program VSE/Access Control Logging and Reporting (ACLR) has been installed. ACLR does not support CICS® sign-on and transaction logging. Logging of CICS sign-on events and accesses to CICS transactions are done on the console.

If **NO** is specified, no checking takes place. The CICS sign-on and transaction security may be active, however.

**NOTAPE** allows to restrict security checking to DASD files and libraries. Tape handling will be the same as for SEC=NO.

**RECOVER** prevents the activation of a security manager (regardless of the SYS ESM= specification). It should only be used for recovery actions that cannot be carried out while a security manager is active.

#### **SERVPART={F1 | F2 | ...FB}**

Specifies the static partition to be used by the security server of the installed security manager. Default is the FB partition.

#### **SPSIZE=nK | mM**

Specifies the size of the storage area to be reserved for shared partitions. The value can be specified either in Kilobytes (K) or in Megabytes (M). If nK is specified, the value is rounded to the next higher multiple of 64. Note that the area may become larger than specified, because the boundary between the shared and the private partition area has to be on a segment boundary (1M).

Valid specifications are:

Maximum: 16M (for the actual maximum values, see "Storage Allocation Rules" on page 38)

Minimum: 0 or - if non-zero value specified - 128K

Default: 0M

If an invalid value is specified for SPSIZE, the system issues a message and assumes the default size.

#### **SUBLIB=m**

Specifies the number of sublibraries which may be attached to the whole VSE system at any time. m must be a decimal integer from 10 to 2000. If the operand is omitted, the system uses the default value of 100. The value m is used to calculate the size of internal library control tables. If the value for m is too small, you may get a control table overflow.

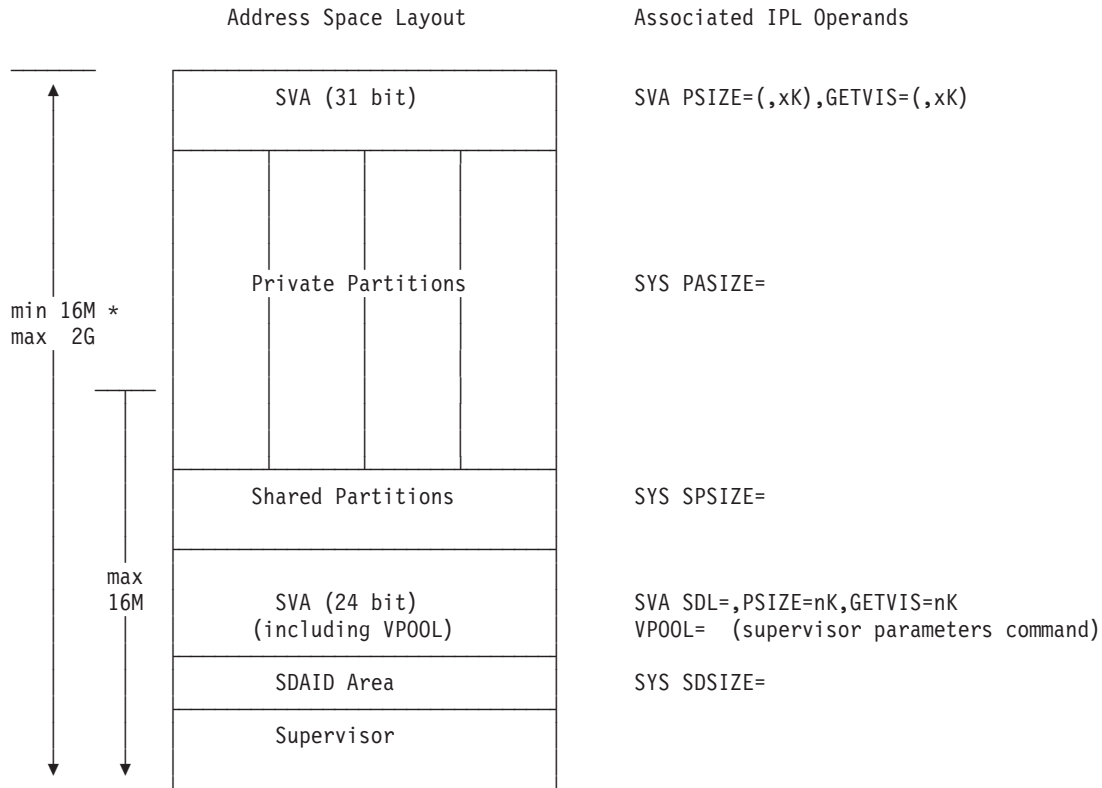
#### **VMCF=YES | NO**

Specifies whether the CMS - VSE console interface is to be activated. The command is ignored if VSE does not run in a VM virtual machine. The default value is VMCF=YES if VSE is a VM guest and VMCF=NO for native VSE.

## Storage Allocation Rules

Figure 2 illustrates the storage layout and the IPL command operands that directly specify the corresponding storage areas.

Figure 3 on page 39 gives a summary of the maximum/minimum and default IPL storage values.



\* For no-PDS systems the address space may be smaller, depending on the size of your processor storage.

Figure 2. ESA Address Space Layout and IPL Operands Determining the Size of The Address Spaces

The following formula lists the rules that have to be observed when specifying the size of the SVA, the shared areas, and the private area. When the sum of all allocated storage areas does not leave a minimum private area below 16M or exceeds the size of the address space, IPL issues a message and terminates.

$$\text{Supervisor} + \text{SDSIZE} + \text{SPSIZE} + \text{PASIZE} + \text{SVA}(31\text{-bit}) + \text{SVA}(24\text{-bit}) \leq \min(2\text{G}, \text{VSIZE})$$

$$\text{Supervisor} + \text{SDSIZE} + \text{SPSIZE} + \text{SVA}(24\text{-bit}) \leq 15\text{M}$$



S U P E R V I S O R	VSIZE (see Note 1)	maximum minimum default	90G 16M 16M	
	VIO	maximum minimum default	128M VPOOL value VPOOL value	
	VPOOL	maximum minimum default	16M 0K 64K	
S Y S	PASIZE (see Note 2)	maximum minimum default	2048M 6M 6M	
	RSIZE	maximum minimum default	16M 0K 64K	
	SDSIZE	maximum minimum default	256K 0K 64K	
	SPSIZE (see Note 3)	maximum minimum default	16M 0/128K 1M	
S V A	PSIZE	maximum minimum default	24Bit	31Bit
			16M 0K 0K	2048M 0K 0K
	GETVIS	maximum minimum default	24Bit	31Bit
			16M 0K 0K	2048M 0K 0K

Figure 3. IPL Storage Values

**Notes:**

1. 90GB is the theoretical VSIZE maximum with which z/VSE can operate. The possible VSIZE value depends on the number of disk devices available for the page data set, where 15 is the maximum. 15 IBM 3390 Model 3 disk devices, for example, allow a VSIZE maximum of 36GB.
2. 2048MB is the theoretical maximum size of an address space. The storage available for PASIZE, however, is 2048 minus the shared areas.
3. IPL accepts a minimum of 0K. However, if a non-zero value is specified, the minimum is 128K (minimum partition size).

## Device Type Codes

Table 1 contains the device type codes for all devices supported by the VSE system, grouped by device class.

Table 1. Device Type Codes

Device Class	Actual IBM Device	Code
Disk	3380 Direct Access Storage <sup>1</sup>	3380
	3380 Direct Access Storage attached to a controller working in NON-SYNC mode	3380 or ECKD
	3390 Direct Access Storage	ECKD
	3390 (in 3380 track compatibility mode)	ECKD
	9336 Direct Access Storage	FBA <sup>2</sup>
	Virtual Disk	FBAV
	9391 - 9397 RAMAC <sup>®</sup> Array Family	ECKD
	2105 - Enterprise Storage Server	ECKD
	2105 - Enterprise Storage Server FCP attached	FBA

<sup>1</sup> Should be added as ECKD<sup>™</sup> device when attached to an ECKD-capable control unit.

<sup>2</sup> Supported as Generic FBA only.

Switch	9032 ESCON <sup>®</sup> Director	ESCD
	9033 ESCON Director	ESCD
Tape	3420 9-track Magnetic Tape Units	3420T9
	3420 7-track Magnetic Tape Units	3420T7
	3480 Magnetic Tape Subsystem	3480
	3480 with IDRC Feature	3490
	3490 Magnetic Tape Subsystem	3480
	3490 with IDRC Feature	3490E
	3490E Magnetic Tape Subsystem	3490
	3590 TotalStorage Enterprise Magnetic Tape Units	TPA
3592 TotalStorage Enterprise Magnetic Tape Units	TPA	

## Device Type Codes

Printers	1403 Printer	1403
	1403 Printer with UCS feature	1403U
	3200 Laser Beam Printer (Supports 8000 Kanji Character Set)	3800
	3211, 3203-5, 3289 Model 4, and 3262 Models 1, 5, 11 printers	PRT1 <sup>2</sup>
	3800 Printing Subsystem	3800
	3800 Printing Subsystem with Burster-Trimmer- Stacker (BTS)	3800B
	3800 Printing Subsystem with BTS and additional Character Generation Storage (CGS)	3800BC
	3800 Printing Subsystem with additional Character Generation Storage (CGS)	3800C
	3800 Model 3 Channel Attached Page Printer	AFP <sup>1</sup>
	3820 Page Printer, emulated (via EML) as 3791L	3791L <sup>1</sup>
	3825 Advanced Function Printer	AFP <sup>1</sup>
	3827 and 3835 Advanced Function Printers	AFP <sup>1</sup>
	3828 Page Printer	AFP <sup>1</sup>
	3831 Page Printer (Japan or Asia Pacific only)	AFP <sup>1</sup>
	3900 Page Printer	AFP <sup>1</sup>
	4245 Line Printer	PRT1 <sup>2</sup>
	4248 Line Printer in 3211 compatibility mode	PRT1 <sup>2</sup>
	4248 Line Printer in native mode	4248
	6262 Models 0xx Line Printers	4248
	Note: All AFP™ capable page printers are added with device type code AFP.	

<sup>1</sup> These device type codes can only be used in the IPL ADD command.

<sup>2</sup> If a PRT1 printer is not attached during IPL, but is later used, the system assumes a 3211 as default printer.

Card Punches	1442N2 Card Punch	1442N2
	2520B2 Card Punch	2520B2
	2520B3 Card Punch	2520B3
	2540 Card Punch	2540P
	3525 Card Punch	3525P
Card Reader	2540 Card Reader	2540R
	3505 Card Reader	3505
Card Read Punches	1442N1 Card Read Punch	1442N1
	2520B1 Card Read Punch	2520B1
	3525 Card Punch (with optional read feature)	3525RP
Printer Keyboard	3215 Console Printer Keyboard	1050A
Display Operator Consoles	3277 Model 2 Display Console	3277
	3278 Model 2A Display Console	3277
	3279 Model 2C Color Display Console	3277

## Device Type Codes

Display Stations	3277 or 3278 Model 2A Display Station (attached in byte mode to a Multiplexer channel)	3277
	3277 or 3278 Model 2A Display Station (attached in burst mode to a Multiplexer Channel)	3277B
	3279 Model 2A Color Display Station 8775 Display Terminal (attached via Loop Adapter feature)	3277B 3791L
	Note: All display stations that comply with the 3270 architecture are added as 3277 or - if operating in burst mode - as 3277B.	
Terminal Printers	3262 Model 3 Printer with 3272 Control Unit attached in burst mode to a Multiplexer Channel (mode must be entered as 01) <sup>1</sup>	3277B
	3282 or 3288 Printer with 3272 Control Unit or 3274-x1B Control Unit When attached in burst mode to a Multiplexer Channel (mode must be entered as 01) <sup>1</sup>	3277 3277B
	3284 or 3288 Printer with 3274-x1D Control Unit (mode must be entered as 01) <sup>1</sup>	3277
	3289 Printer (except Model 4) with 3274-x1B Control Unit When attached in burst mode to a Multiplexer Channel (mode must be entered as 01) <sup>1</sup>	3277 3277B
	4019 Desk Top Page Printers Supported like a 4028 Laser Printer	3277B
	4028 Model NS1 Laser Printer Supported like a 3263 Model 3 Printer	3277B
	4029 Laser Printer Supported like a 4028 Laser Printer	3277B
	4230 Printer Supported like a 4224 Printer	3277
	6252 Models Dxx Printer Supported like a 3262 Model 3 Printer	3277B
	6262 Models Dxx Printer Supported like a 3262 Model 3 Printer	3277B

<sup>1</sup>These terminal printers cannot be assigned to SYSLST.

## Device Type Codes

Communication Control Units	3172 Interconnect Controller 3174-L Control Unit 3174-xL Control Unit 3274-x1A Control Unit 3274-x1B Control Unit 3274-x1D Control Unit 3745 Communications Controller 3791 Local Communications Controller Channel-to-Channel Adapter Open Systems Adapter 2 and OSA Express in non-QDIO mode OSA Device (to be used by OSA/SF) OSA Express (QDIO mode) HiperSockets FCP Adapter  Note: All local SNA terminal controllers are added with device type code 3791L.	CTCA 3791L 3277 3277 3745 3791L CTCA OSA OSA OSAD OSAX OSAX FCP  3791L
Unsupported Devices	Unsupported, no burst mode on multiplexer channel Unsupported, with burst mode on multiplexer channel	UNSP UNSPB

## Device Type Codes

## Chapter 3. Job Control and Attention Routine

This section contains descriptions, formats, and usages of the job control commands and statements, and attention (and other system) routine commands, which are identified as follows:

- Job control statement - JCS
- Job control command - JCC
- Attention routine command - AR

**Note:** The characters 'AR' stand for attention routine **and other system routine** commands.

Table 2 contains the commands and statements grouped by function, and also indicates the programs or routines for which they are valid. Under the columns **AR** and **JCC**, letter combinations **XR** and **XS** indicate whether a command requires unrestricted command authorization (XR) or is semi-restricted (XS). For a detailed description of *command authorization* in z/VSE refer to the z/VSE *Operation* under "VSE System Control Commands".

A few commands show different degrees of restriction between JCL and AR type. This has either to do with different levels of control that JCL and AR routines maintain in a given situation (CANCEL command, for example). Or, JCL and AR commands may have different scopes of function. For example, the **JCL** command LOG only affects the partition in which it is issued whereas the **AR** command LOG affects all static partitions.

Table 3 on page 49 contains a brief description of all job control statements.

Table 2. JCS, JCC, and AR by Function

Type of Command or Statement	Operation	Valid for JCS	Valid for AR	Valid for JCC
Job Identification	Job /&	X X		
User Identification	ID	X		X
File Definition	DLBL EXTENT TLBL /* /+	X X X X X		
Library Definition	LIBDEF LIBDROP LIBLIST	X X X		X X X
Pass Information to Operator	*	X		

## JOB CONTROL / ATTENTION ROUTINE

Table 2. JCS, JCC, and AR by Function (continued)

Type of Command or Statement	Operation	Valid for JCS	Valid for AR	Valid for JCC
Job Stream Control	BATCH CANCEL PAUSE PRTY START STOP TPBAL UNBATCH	X	XR XS XS XS XR  XS	X X XR XR XR  XR (see note)
Setting Symbolic Parameters	SETPARM PROC	X X		X X
Conditional Job Control	/. GOTO IF ON	X X X X		X X X
Setting System Parameters	ALLOC EXPLAIN MSECS NPGR SET SIZE STDOPT SYSDEF VDISK	X X X	XR XS XS  XR  XR	XR X XR XR XR XR XR XR X
Pass Information to Program	DATE OPTION UPSI	x x x		
Execution of Program	EXEC RSTRT	X X		X



## JOB CONTROL / ATTENTION ROUTINE

Table 2. JCS, JCC, and AR by Function (continued)

Type of Command or Statement	Operation	Valid for JCS	Valid for AR	Valid for JCC
Operator Communications	ALTER		XR	
	CACHE		XS	
	DSPLY		XR	
	DUMP		XR	
	END or ENTER key		X	X
	GETVIS		X	
	HCLOG		XS	
	IGNORE		XR	XR
	LIBSERV	X	XR	X
	LOG		XR	X
	MAP		X	X
	MSG		XS	
	NEWVOL		XR	
	NOLOG		XR	
	OFFLINE		XR	
	ONLINE		XR	
	OPERATE		XR	
	PRTYIO		XS	
	PWROFF		XR	
	QUERY	X	X	X
	RC		XR	
	REDISPLAY		XS	
	REPLID		XS	
STATUS		X		
SYSECHO		XS		
UNLOCK	X	XR		
ZONE				
* CP			XR	XR
Control of I/O System	ASSGN	X		X
	BANDID		XR	
	CLOSE	X		X
	DVCDN			XR
	DVCUP			XR
	FREE		XR	
	HOLD			XR
	JCLEXIT			XR
	LFCB		XR	
	LISTIO	X		X
	LUCB		XR	
	MTC	X	XR	X
	PWR	X		X
	RESERV		XR	
	RESET	X		X
	ROD			XR
	SETDF		XR	
	SETPFIX	X		X
	SERPRT	X		X
	UCS			XR
VOLUME		XR		
VTAPE	X		XR	

**Note:** Valid only in a foreground partition.

### Job Control Statements

Job Control statements must conform to the following formatting rules.

- **Identifier.** Two slashes (//) identify the statement as a control statement. They **must** be in columns 1 and 2. At least one blank must immediately follow the second slash.

**Exceptions:**

- / (label statement)
- /& (end-of-job statement)
- /\* (end-of-data file statement)
- /+ (end-of-procedure statement)
- \* (comment statement)

- **Operation.** Describes the operation to be performed. At least one blank follows its last character.
- **Operands.** May be blank or may contain one or more entries. If a statement includes two or more operands, they must be separated by commas. The last term **must** be followed by a blank, unless its last character is in column 71. Any blank within the operand is considered an end-of-operand indication, and no further processing of that statement occurs, unless the operand is within quotes. Exceptions are the IF and ON statements, which have blanks between the condition expression and the action specification.

---

### Job Control and Attention Routine Commands

Job control commands and attention routine commands contain the operation code, at least one blank, and then the specified operands. The operands are separated by commas. The operation code usually begins in column 1 of the command, but this is not required.

- Job control commands (JCC) are issued between jobs or job steps and are entered through SYSRDR or SYSLOG. (Job control statements, on the other hand, are usually coded as part of the input stream and are entered through SYSRDR.)
- Attention routine commands (AR) can be issued from the console at any time.

## Job Control Statements Summary

A brief description of the job control statements follows.

Table 3. Job Control Statements Summary

ASSGN	Used at execution time to assign a specific device address to the symbolic unit name used.
CLOSE	Closes either a system or a programmer logical unit assigned to tape, disk, or diskette.
DATE	Contains a date that is put in the communications region.
DLBL	Contains file label information for disk or diskette label checking and creation.
EXEC	Indicates the end of job control statements for a job step and that the job step is to be executed.
EXEC PROC / EXEC REXX	Calls a cataloged procedure and defines values for symbolic parameters.
EXTENT	Defines each area, or extent, of a disk file or diskette volume.
GOTO	Causes JC to skip all following statements (except JOB, /&, /+) up to the specified label statement.
ID	Used to specify user identification and password.
IF	Causes skipping or execution of the following statement dependent on the specified condition.
JCLEXIT	Activates or deactivates one or more JCL exit routines.
JOB	Indicates the beginning of control information for a job.
LIBDEF	Defines library chains.
LIBDROP	Drops library chain definitions.
LIBLIST	Lists library chain definitions.
LIBSERV	Controls 3494 tape system.
LISTIO	Used to get a listing of I/O assignments on SYSLOG or SYSLST.
MTC	Controls operations on magnetic tapes.
ON	Causes specified action to be done if the specified condition is true after any step in the following job stream.
OPTION	Sets one or more of the job control options.
PAUSE	Causes a pause immediately after processing this statement, or at the end of the current job step.
PROC	Defines and initializes symbolic parameters in a procedure.
PWR	Passes a PRELEASE or PHOLD command to POWER.
QUERY / QUERY TD	Displays information on data spaces, standard options, and Multiprocessor environment.
RESET	Resets I/O assignments to the standard assignments.
RSTRT	Restarts a checkpointed program.
SETPARM	Assigns a character string or return code to the specified parameter.
SETPFIX	Defines limits for PFIxing pages.
SETPRT	Loads the IBM 3800 buffers.
STDOPT	Resets system defaults.
SYSDEF / SYSDEF TD	Defines limits and defaults for data spaces. Enables Multiprocessor environment.
SYSDEF SCSI	Associates VSE SCSI device number (FBA) with real SCSI Logical Unit Number (LUN), and its access path (FCP, WWPN).
TLBL	Contains file label information for tape label checking and writing.
UPSI	(User Program Switch Indicators) Allows the user to set program switches that can be tested.
VDISK	Defines the layout of a virtual disk.
VTAPE START	Defines the dataset to contain the virtual tape
VTAPE STOP	Releases the access to the virtual tape
ZONE	Initializes the zone field in the communications region.
/.	Label statement.
/*	Indicates the end of a data file.

## JOB CONTROL / ATTENTION ROUTINE

Table 3. Job Control Statements Summary (continued)

/&	Indicates the end of a job.
*	Job control comments.
/+	Indicates the end of a procedure or librarian End-of-Data.

If an invalid job control statement is entered, a message is issued so that the programmer or operator can correct the statement in error.

Whenever an invalid statement is indicated, the entire statement must be reissued to be effective. This rule applies even if only one operand was invalid. It also applies if the statement itself was correct, but could not be executed because the appropriate system environment was not available. For example, if an OPTION LINK is entered without a SYSLNK assignment, the OPTION statement is invalid. You must re-enter the OPTION statement after assigning SYSLNK.

---

### Sequence of JCS and JCC

The job control statements for a specific **job** always begin with a JOB statement and end with a /& (end-of-job) statement. A specific job consists of one or more **job steps**. Each job step is initiated by an EXEC statement. Preceding the EXEC statement are any job control statements necessary to prepare for the execution of the specific job step. One limitation on the sequence of statements preceding the EXEC statement is that DLBL statements must immediately precede the corresponding EXTENT statements. If the DLBL and EXTENT statements for a temporary SYSLNK area are in the job stream, they should precede the OPTION LINK or OPTION CATAL statement.

---

### Conditional Job Control

Normally, the statements or commands in a job stream are read by job control in the sequence in which they are entered on SYSRDR or SYSLOG, and the job steps are executed in this order.

However, you can cause the system to execute or bypass parts of the job stream conditionally, dependent on the result of previously executed steps within the same job.

The result of a job step can be reflected in a return code between 0 and 4095, which must be set by the executed program

- By placing the desired return code in register 15 and branching at the end of the program to the return address which was supplied in register 14 when the program was invoked. The programmer must take care that register 15 is set correctly, otherwise the job flow will be unpredictable. (Note that the first two bytes of Register 15 are not part of the return code. Bit 0 of the register indicates whether a dump is required, the rest of the two bytes is reserved.)
- By the EOJ macro
- By the DUMP macro
- By an equivalent method in high-level programming languages.

If a return code greater than 4095 is issued, job control assumes a return code of 4095. If no return code is issued, a return code of zero is assumed. The return code can be tested (see statements IF, ON), and the sequence of execution altered if

appropriate (see statements GOTO, /. label), or the step parameters for a following step can be set accordingly (see statement SETPARM). If the job control program receives a return code greater than or equal to 16, it terminates the job, unless an ON statement specifying a different action for this return code has been given. Such an action may be the execution of a certain part of the job stream after abnormal termination or cancelation of the job. The actions to be taken in the case of abnormal termination must be specified in the statements:

- ON \$RC>16... if a high return code is encountered;
- ON \$ABEND... if the job terminates abnormally; or
- ON \$CANCEL... if the job is canceled.

---

### Parameterized Procedures

In order to make the handling of job streams easier and more flexible, z/VSE provides facilities for altering, not only the sequence of execution of job control statements, but also the values in their operands.

The value, or part of the value, in the operand field of a job control statement can be modified at execution time if it is coded as a symbolic parameter (see below), and the appropriate new value is assigned to it

- In a PROC statement (if the parameter is in a cataloged procedure)
- In a // EXEC PROC statement (if the parameter is in a cataloged procedure)
- By a SETPARM command or statement which precedes it in the job stream
- By a SETPARM command or statement entered by the operator.

The value you assign to a symbolic parameter must be a character string. It can be 0 to 50 characters long. If the string contains national or special characters, it must be enclosed in quotes, which the system does not take as part of the value of the string. A string consisting of only alphabetic and/or numeric characters does not need enclosing quotes. Quotes may not be used within the string itself. An ampersand (&) within the string must be coded as a double ampersand (&&) to avoid confusion with the delimiters of symbolic parameters.

The current value of a symbolic parameter can also be tested in an IF statement, giving you the possibility of influencing the sequence of execution of a job.

---

### Symbolic Parameters

A symbolic parameter is a name consisting of one to seven alphanumeric characters, of which the first must be alphabetic. This name may be freely chosen, and the same symbolic parameter may occur any number of times in a job stream.

Avoid using OV as a symbolic parameter name, since it is internally used by Job Control.

When a symbolic parameter is used in the operand of a job control command or statement you must indicate to job control that it is a symbolic parameter by preceding it with an ampersand (&). The end of the parameter must be indicated by a period (.), unless it is followed by a delimiter, that is, a nonalphanumeric character.

The operation and comments fields of job control commands or statements may not contain symbolic parameters, only the operand field. However, symbolic

## JOB CONTROL / ATTENTION ROUTINE

parameters may occur anywhere in the operand field. This includes operands in quotes, for example the file ID in DLBL statements or the PARM operand in EXEC PROC statements.

Here is an example of an ASSGN statement with symbolic parameters:

```
// ASSGN SYS001,&UNIT.&VOLUME.
```

Here, you could omit the periods, like this:

```
// ASSGN SYS001,&UNIT&VOLUME
```

because the symbolic parameters are ended by the nonalphameric characters '&' and blank, respectively. Let us assume that this statement is contained in a cataloged procedure named PROC1, and that SYS001 should normally be assigned to the physical unit address 380, and that you do not require any special tape volume. You should code the procedure PROC1 as follows:

```
// PROC UNIT=380,VOLUME=  
...  
...  
// ASSGN SYS001,&UNIT&VOLUME
```

When the procedure is called, job control will substitute 380 for &UNIT, and ignore the VOLUME parameter. The job will be executed as if the statement read:

```
// ASSGN SYS001,380
```

If, for a particular run of the application, you want to use the device address 381, and use a particular tape volume, for example 888888, you must assign the appropriate values in the // EXEC PROC statement, as follows:

```
// EXEC PROC=PROC1,UNIT=381,VOLUME=',VOL=888888'
```

The job will then be executed as if the statement read:

```
// ASSGN SYS001,381,VOL=888888
```

That is, the values given in the EXEC PROC statement override those in the PROC statement.

You may want to assign SYS001 to unit 382 and use volume 777777 if, for example, the preceding job step ended with a return code higher than 4. In this case, you would code your procedure PROC1 as follows:

```
// PROC UNIT=380,VOLUME=  
...  
...  
// IF $RC>4 THEN  
// SETPARM UNIT=382,VOLUME=',VOL=777777'  
// ASSGN SYS001,&UNIT&VOLUME  
...
```

If the preceding step does end with a return code greater than 4, the job will be executed as if the statement read:

```
// ASSGN SYS001,382,VOL=777777
```

That is, the values assigned in the SETPARM statement override those given in the PROC and EXEC PROC statements. A job control operand may consist of several symbolic parameters, or it may consist partly of a symbolic parameter, partly of a literal specification. The value assigned to the symbolic part is concatenated with the literal part, as follows:

PARAMETER	ASSIGNMENT	EXECUTED AS
-----------	------------	-------------

&SIZE.(80)	SIZE=BLOCK	BLOCK(80)
&SIZE(80)	SIZE=BLOCK	BLOCK(80)
&LIBNAME.LIB	LIBNAME=PRIV	PRIVLIB
SYS&NUM	NUM=003	SYS003
&A..B	A=X	X.B
&C&UU	C=2,UU=81	281

You may also assign to a symbolic parameter a value consisting of several job control statement operands, as follows:

PARAMETER	ASSIGNMENT	EXECUTED AS
&OPERAND	OPERAND='380,VOL=666666'	380,VOL=666666

---

## Nested Procedures

A cataloged procedure can call another cataloged procedure. That is, an EXEC PROC statement may occur within a procedure. Procedure A “contains” procedure B if the statement EXEC PROC=B occurs in procedure A. If the statement EXEC PROC=C occurs in procedure B, then procedure B contains procedure C, and procedure A also contains procedure C. Or conversely:

- Procedure B is contained in procedure A;
- Procedure C is contained in procedure A and in procedure B.

In general, any cataloged procedure may call any other cataloged procedure, with the following exceptions:

- A procedure must not call itself. That is, procedure A must not issue the statement EXEC PROC=A.
- A procedure must not call a procedure in which it is contained. That means, in the example above, that procedure B must not issue the statement EXEC PROC=A, and procedure C must not issue the statements EXEC PROC=A or EXEC PROC=B.
- All procedures in one nesting must have been cataloged with the same DATA= operand on the CATALOG statement (either all with DATA=YES or all with DATA=NO).

Procedures can be nested at up to 16 levels. Nesting Level 0 denotes the job control statements read from SYSRDR or SYSLOG. Level 1 denotes procedures called by an EXEC PROC statement on SYSRDR or SYSLOG. Level 2 denotes procedures called from Level 1 procedures, and so on up to Level 15. In the example above, assuming that EXEC PROC=A was issued from SYSRDR, procedure A is Level 1, procedure B is Level 2 and procedure C is Level 3.

---

## Scope of Symbolic Parameters

A symbolic parameter is normally valid only at the nesting level on which it is defined. If defined by a SETPARM statement issued from SYSRDR, the parameter is valid until End-of-Job. If the SETPARM statement is in a procedure, the parameter it defines is valid until End-of-Procedure.

If a parameter is defined in an EXEC PROC or PROC statement, it is valid until End-of-Procedure.

Note that a parameter can be passed to a lower-level procedure, for example from a Level 1 procedure to a Level 2 procedure. The parameter will then remain valid after the lower-level procedure ends, but will cease to be valid at the end of the procedure in which it was defined.

## JOB CONTROL / ATTENTION ROUTINE

For detailed information on nested procedures and the passing and substitution of parameters, see "Using Nested Procedures" in the *z/VSE Guide to System Functions*.

---

### Printing of Job Control Statements and Commands

Job control statements and commands are printed on SYSLOG and/or SYSLST after they have been processed, as follows:

- Symbolic parameters are substituted
- The active data in columns 16 - 71 of continuation cards are chained together
- Unnecessary blanks are removed, where this will not affect performance
- Double ampersands and quotes are reduced to single ones
- Continuation cards which were not processed by JC because they were in error are printed in their original format to facilitate debugging
- Statements that have become longer than 120 characters (because of chaining of continuation cards) are printed line by line (120 characters per line).

If you wish to have your JC statements or commands printed out in the format in which you coded them, you can specify the operand LOGSRC in the OPTION statement of the job. In this case, the system will write each statement which contains symbolic parameters or a continuation twice, once in the source form, as coded, and once in the form described above.

---

### Command Authorization in Job Control

Several Job Control commands are restricted to users with master authority. In this context a "user with master authority" is

- either administrator (user type 1 in the user profile)
- or programmer (user type 2 in the user profile) authorized to access a master console.

If a restricted Job Control command is entered from a user console, the message "COMMAND NOT ALLOWED, INSUFFICIENT AUTHORITY" is displayed, regardless of whether access control checking (secured system) is active or not or whether an ID statement was given beforehand.

If such a command is entered through SYSRDR and access control checking is not active, the command is accepted and executed. If however access control checking is active (secured system) the command is rejected, unless there was a preceding ID command or statement identifying a user with administrator or master console authorization.

Authorization is required for the following Job Control commands:

ALLOC  
DVCDN  
DVCUP  
HOLD  
JCLEXIT  
MSECS  
NPGR  
PRTY  
ROD  
SET  
SIZE  
START



STOP  
 SYSDEF  
 UCS  
 UNBATCH  
 VTAPE  
 \* CP

---

## Command Authorization in Attention Routine (AR)

Similar restrictions hold for AR commands, which can be classified into three categories:

- Restricted commands that are supported only from system and master consoles.
- Semi-restricted commands that are supported from user consoles only for certain argument values. The most common case are partition related commands, that are accepted only when an ECHO option for the originating console is effective for the job currently running in the specified partition (ECHO scope).  
 Another case are commands that are accepted from user consoles only in a “query” form.
- Commands for general use that may be entered from any console.

Since most AR commands are restricted, only those classified as semi-restricted or for general use are listed below.

CANCEL	ECHO scope
GETVIS	General use
MAP	General use
MSG	ECHO scope
MSECS	Query only
PAUSE	ECHO scope
PRTY	Query only
PRTYIO	Query only
QUERY	General use
STATUS	General use
TPBAL	Query only
VOLUME	General use

If a restricted AR command is entered from a user console, the message “COMMAND AUTHORIZATION INSUFFICIENT” is displayed.

---

## Command and Statement Formats (JCS, JCC, AR)

Detailed descriptions of the formats and functions of individual JCS, JCC, and AR statements and commands follow in alphabetic order. If the JCS and JCC or JCC and AR formats coincide, this is indicated under “Type”.

## ALLOC (Allocate Storage to Partitions)

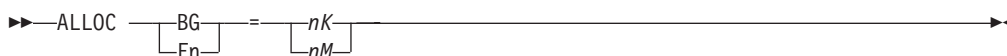
The ALLOC command allocates virtual and processor storage to the static partitions of a VSE system.

The command is not allowed in a dynamic partition.

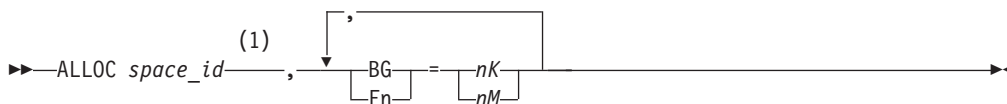
The layout of virtual storage is described in “Chapter 1. Storage Management” in the *z/VSE Guide to System Functions*.

The command can be given in four different formats:

### Format 1 (JCC, AR)



### Format 2 (JCC, AR)



#### Notes:

- 1 Valid space-IDs are 0-9, A, and B

### Format 3 (JCC, AR)



### Format 4 (JCC, AR)



### Format 1

The first command format should be used if only a single partition is to be allocated in an address space (in the following this format is called ‘single-partition allocation’).

The requested partition size determines the size of the allocated storage in the partition’s address space. With single-partition allocation, virtual storage is saved, since page management tables for the new address space are created according to the partition size and not to the size of the whole private area (SYS PASIZE).

If a single partition is to be allocated

- The maximum amount of virtual storage that may be allocated to the partition is defined by the PASIZE value specified in the IPL SYS command.

- Reallocation must also be done with single-partition allocation (except when the partition is deallocated first, that is, set to 0K).
- Reallocation must not increase the initial allocation size (if an increase is required, the partition must be deallocated first). As an exception, the background (BG) partition may be increased up to the value of PASIZE.

Since z/VSE supports 12 address spaces (plus S and R), each static partition will be allocated within its own **default** address space:

Partition	BG	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB
Space-ID	0	1	2	3	4	5	6	7	8	9	A	B

**Note:** With single-partition allocation, you cannot allocate a new partition if the related address space already exists (created by the second command format).

### Format 2

With the second command format, more than one partition may be allocated within an address space (in the following this format is called '**multiple-partition allocation**').

Supported space-ids are 0,1,..9,A,B.

- The maximum amount of virtual storage that may be allocated to a private static partition in an address space is defined by: PASIZE minus the size of any partition(s) already allocated in the address space.
- Several partitions may be allocated within an address space. However, only the **last** one of these may cross the 16MB line. Allocation of this partition is performed only when at least 128KB (that is, a minimum partition size of 80KB and a minimum GETVIS size of 48KB) can be allocated below the 16MB line.
- If a partition has been allocated with multiple-partition allocation, it must be reallocated with multiple-partition allocation, too (except when it has been deallocated to 0K before). The partition size can, however, be increased.
- With multiple-partition allocation, you may allocate new partitions into an existing address space, provided this address space has also been created with multiple-partition allocation.

### Format 3

With the third command format, **real** storage is allocated to static partitions.

Allocating real storage with the ALLOC R,... command requires a specification of RSIZE=nK in the IPL SYS command. This means that you have to specify RSIZE during IPL whenever you plan to use an ALLOC R,... command. RSIZE specifies the total amount of real storage that may be allocated for all static partitions (as a sum) with ALLOC R.

ALLOC R (and RSIZE) should be used only when real execution of programs (with EXEC program,REAL) is required. If real storage is needed for PFXing, the SETPFX command should be used.

**Note:** An ALLOC R command for a given partition can be issued only after virtual storage has been allocated for the same partition.

### Format 4

With the fourth command format, static partitions can be allocated within the shared virtual address space.

## Description of Operands

### BG | Fn

Indicates the partition to which storage is to be allocated. Valid specifications are:

BG, F1 through F9, FA, FB.

At least one partition must be specified. If you want to change the size of the BG partition, you cannot use a space-id other than 0, since the BG partition has already been allocated during IPL and cannot be deallocated.

### nK

Specifies, in Kilobytes, the amount of storage to be allocated. Valid specifications are 0 or an integer. Specifying 0 means that the entire storage allocated to the named partition is freed. The partition can no longer be used.

The system rounds up the specified integer to a multiple of:

64 for virtual address spaces

4 for real address space

The resulting rounded-up value must be at least

4K in real address space allocations, and

128K in virtual address space allocations.

### nM

Specifies, in Megabytes, the amount of storage to be allocated. Valid specifications are 0 or an integer. Specifying 0 means that the entire storage allocated to the named partition is freed. The partition can no longer be used.

### space\_id

Indicates in which address space the specified amount of storage for the named partition(s) is to be allocated. Valid specifications are: 0 through 9, A, B.

**R** Indicates the real address space (processor storage).

**S** Indicates the shared virtual address space.

After IPL, the BG partition has a default size of 1 megabyte. Partitions must always be allocated or reallocated explicitly; the size of an unspecified partition is **not** changed. To delete a foreground partition from the system, you must issue an UNBATCH command in the partition, and then an ALLOC command specifying a size of 0K or 0M for the respective partition. No allocation takes place when the ALLOC command would move the start address of a partition upward and/or the end address downward while that partition is active. (Exception: The end address of the partition in which job control is processing an ALLOC command may be moved downward as long as the virtual storage allocated to the partition does not drop below 128K.)

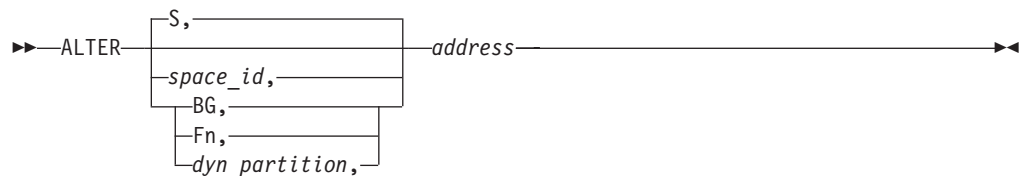
## ALTER (Alter Contents of Virtual Storage)

The ALTER command allows the operator to alter 1 to 16 bytes of virtual storage, starting at the specified hexadecimal address. After the command has been entered and the END/ENTER key pressed, the hexadecimal representation of the information to be placed in storage should be entered on the device assigned to SYSLOG. Two hexadecimal characters (0 through F) must be entered for each byte to be changed. If an odd number of characters is entered, the last character is ignored and its associated byte is unaltered.

The storage bytes to be altered will be displayed before the operator can actually change them.

**Note:** The ALTER command does not work in a multiprocessor environment for addresses lower than x'1000'. See message 1I37I for a detailed explanation.

### AR Format



#### space\_id

Specifies in which address space the alteration at the given address is to be made. Valid specifications are:

- R (real) or S (shared)
- 0 through 9, A, B

The default value is S.

#### BG | Fn | dyn\_partition

Specifies in which static or dynamic partition the alteration at the given address is to be made. You can specify any of the static partitions BG, F1 through FB or a partition within a dynamic class, for example, P1.

#### address

Indicates the address at which storage alteration is to start. **address** can be a 1- to 8-digit hexadecimal address. The highest address that can be specified is limited by the size of the shared areas plus the size of the private area (SYS PASIZE value).

If space-id S has been specified and the specified address is not within the shared area (supervisor, SVA or shared partitions), the command is ignored and a corresponding information message is issued. If the specified space-id is one of 0 through 9, A or B, any address between 0 and end-of-storage is accepted.

If the specified address is within a dynamic partition, the corresponding dynamic space GETVIS area can be altered, too.

If the specified address is within an address range which has not been allocated to a partition, the command is ignored and a corresponding information message issued.

If the bytes to be altered cross the boundary from a valid to an invalid address area, only the bytes in the valid area are altered and a corresponding information message is issued.

## ASSGN (Assign Logical Unit)

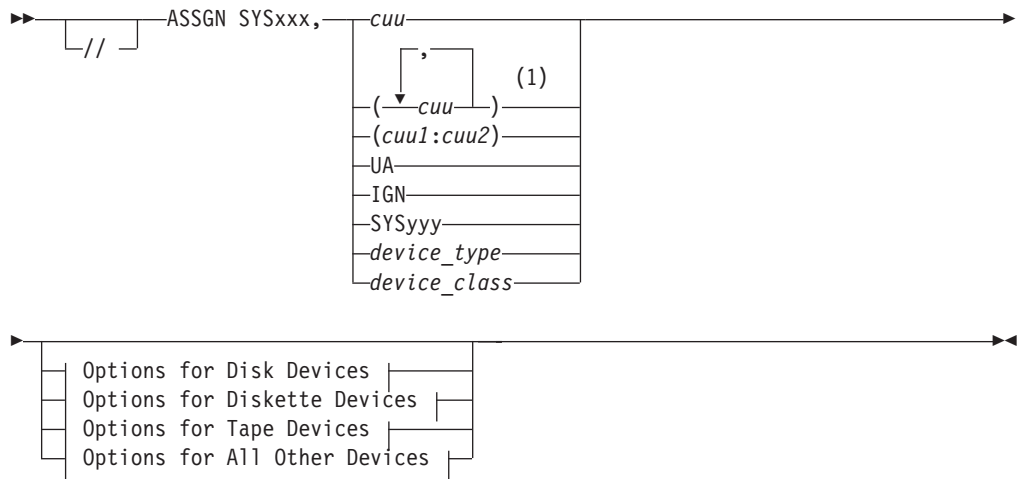
The ASSGN command or statement assigns a logical I/O unit to a physical device. Multiple logical units are allowed to be assigned to one physical unit within the same partition. Assignments are effective only for the partition in which they are issued. No physical devices except disks can be assigned to (shared by) several active partitions at the same time.

Continuation lines are accepted for the ASSGN command or statement.

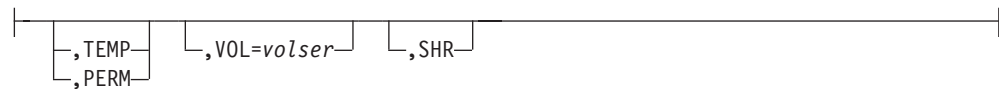
The job control **statement** (type JCS) is temporary. It remains in effect only until the next change in assignment or until the end of job, whichever occurs first. At the completion of a job, a temporary assignment is automatically restored to the permanent assignment for the logical unit.

The job control **command** (type JCC) is permanent. It remains in effect until the next permanent assignment, a DVCDN command, or re-IPL of the system, whichever occurs first. A CLOSE command for a system logical unit on disk or diskette also removes a permanent assignment. See also the description of the TEMP/PERM operands.

### JCS, JCC Format



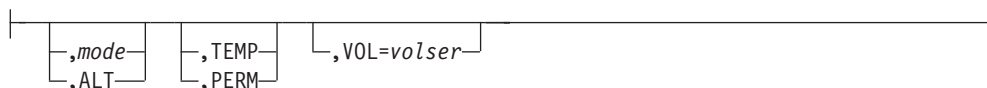
#### Options for Disk Devices:



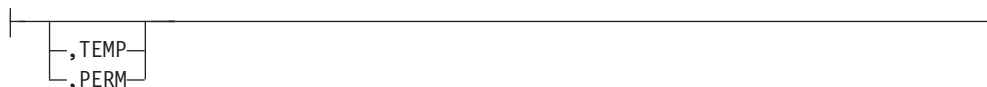
#### Options for Diskette Devices:



#### Options for Tape Devices:



### Options for All Other Devices:



### Notes:

- Up to 64 device addresses can be specified

**Note:** If you use two or more optional operands in an ASSGN statement, they must be entered in the sequence shown here.

### SYSxxx

Represents the logical unit name. It can be either:

- One of the following *system* logical unit names:
  - SYSRDR
  - SYSIPT
  - SYSIN
  - SYSPCH
  - SYSLST
  - SYSOUT
  - SYSLNK
  - SYSLOG
- Or a *programmer* logical unit SYSnnn, where nnn can be a decimal number from 000 to 254.

SYSCAT and SYSREC can only be assigned with the DEF command during IPL. For compatibility reasons, an assignment for SYSREC entered from SYSRDR is ignored and processing continues; if entered from SYSLOG, the assignment is rejected.

**Restrictions:** The type of device assignment is restricted under certain conditions:

- If you want to make an assignment for the operator console (ASSGN SYSLOG, cuu) or if you want to assign a programmer logical unit to SYSLOG (ASSGN SYSnnn, SYSLOG) and the console is running in disconnected mode (OPERATE DISC), the console cannot be assigned and message UNIT CURRENTLY UNASSIGNABLE will be issued.
- If one of the system logical units SYSRDR, SYSIPT, SYSLST or SYSPCH is assigned to a disk device or diskette, the assignment must be permanent and follow the DLBL and EXTENT statements.
- If SYSRDR and SYSIPT are to be assigned to the same disk device or diskette, SYSIN must instead be assigned and this assignment must be permanent.
- SYSOUT is only valid for a tape unit and must be assigned permanently.
- SYSLOG can only be assigned permanently.
- If SYSIPT is assigned to a tape unit, it should be a single file and a single volume.

## ASSGN

7. You may not assign SYSLOG to a 3278 Model 2A or 3279 Model 2C with a message area of 16 lines if IPL was done from a 3277, 3278 or 3279 with a message area of 20 lines.
8. SYSLOG cannot be assigned to a console printer (3284, 3286, 3287, 3288).
9. ASSGN SYSLOG,UA and ASSGN SYSLOG,IGN are not accepted.
10. If a system logical unit is assigned to a tape, disk, or diskette, the unit must be closed (using the CLOSE command) before it can be reassigned.
11. When SYSOUT is assigned to a magnetic tape device it must not be the permanent assignment of either SYSLST or SYSPCH. Before assigning a tape drive to a system output unit (SYSOUT, SYSLST, SYSPCH), all previous assignments of this tape drive to any system input units and to any programmer units (input or output) must be permanently unassigned. The assignment of SYSOUT must always be permanent.  
Also, before assigning a tape to a system input unit or any programmer unit, all previous assignments of this tape to any system output unit must be permanently unassigned.
12. A programmer logical unit cannot be assigned to SYSLST if SYSLST has been assigned to tape or disk before.
13. ASSGN SYSRDR and ASSGN SYSIPT are allowed within a cataloged procedure. SYSRDR assignments, and SYSIPT assignments in a procedure with a DATA=YES specification, become effective on returning to JC level 0. SYSIPT assignments in a procedure with DATA=NO specification become effective immediately. (In a cataloged procedure, to read data on SYSIPT, you must use a DTFDI instead of a DTFCD.)
14. In a system with 16 channels, ASSGN SYSxxx,FBA cannot be used to address unit BA on channel F. Use ASSGN SYSxxx,X'FBA' to distinguish the cuu specification from the device class specification FBA (for Fixed Block Architecture).

### cuu | X'cuu'

Indicates the physical unit address to which the specified logical unit is to be assigned.

c = channel number  
uu = unit number

The form X'cuu' must be used when the physical address of the specified device is FBA (that is, channel F, unit BA), to distinguish it from the device class FBA (for Fixed Block Architecture disk unit). Otherwise, the X' ' can be omitted.

When you assign a 4248 printer using the physical unit address, you must check whether it is in the mode you require. Programs can use this printer in native mode only if it was added with the device-type 4248 at IPL. 4248 printers in 3211 mode and added as PRT1 cannot be used in native mode.

### (cuu,cuu,...,cuu)

You can specify a list of up to 64 device addresses, enclosed in parentheses. In this case the system searches only the addresses specified in the address list for an unassigned unit, starting with the first specified device address. Once a free unit is found, it is assigned to SYSxxx for the job in which the assignment is made.

**Note:** If the address list contains the addresses of a 3480 Tape Subsystem and another tape unit, do **not** specify a mode setting (**mode** operand) in the ASSGN statement.



For disks, if SHR is specified, the first unit in the list is assigned, even if previously assigned. (See Table 5 on page 66.)

**(cuu1:cuu2)**

The format (cuu1:cuu2) specifies a list of device addresses starting with cuu1 and ending with cuu2. For example

```
ASSGN SYS005,(200:202)
```

is equivalent to

```
ASSGN SYS005,(200,201,202)
```

Note that cuu2 must be greater or equal than cuu1, and it must be less than cuu1+x'40'.

**UA**

Indicates that the logical unit is to be unassigned. Any operation attempted on an unassigned device cancels the job.

**IGN**

For certain American National Standard and DOS/VS COBOL application programs (for sequential input files), and for FORTRAN, the IGN option unassigns the specified logical unit, and ignores any subsequent logical IOCS command (OPEN, GET, etc.), issued for that unit. This allows you to disable a logical unit that is used in a program without removing the code for that unit. You can then execute the program as if the unit did not exist. This may be especially helpful when debugging a program.

For assembler language application programs, IGN indicates that the logical unit is to be ignored. With files processed by logical IOCS, the OPEN to the file is ignored, the DTF table is not initialized (for example, IOREG, extent limits), and the IGNORE indicator is set on in the DTF table. It is your responsibility to check this indicator and bypass any I/O commands (GET, PUT, etc.) for this file.

The IGN option is not valid for SYSRDR, SYSIPT, or SYSIN, nor for PL/I programs. The IGN option can be made temporary by specifying the TEMP option.

When using ASSGN IGN for associated files, all logical units of the associated files must be assigned IGN.

Additional information about IGN is in the "OPEN(R)" section of *z/VSE System Macros Reference*. IGN restrictions for users of American National Standard and DOS/VS COBOL and of RPG II\* are given in the associated Program Product publications for the compiler being used.

**SYSyyy**

This may be any system or programmer logical unit, except SYSCAT (see SYSxxx, above). If this operand is specified, SYSxxx is assigned to the same device to which SYSyyy is currently assigned. This type of specification is particularly helpful because the specification of SYSxxx,SYSyyy is considerably shorter than the full specification.

Examples:

```
// ASSGN SYS001,3380,PERM,VOL=RAFT01,SHR
// ASSGN SYS003,SYS001
// ASSGN SYSLNK,SYS001
```

**device\_class**

This type of specification can be used if the exact configuration of the installation is not known or not important. It allows to specify a generic name

## ASSGN

like PRINTER, TAPE, DISK for the devices listed below. The system searches for the first unassigned unit within the specified device-class and assigns it to SYSxxx (see Table 5 on page 66).

The device class FBAV indicates that a virtual disk is to be selected for assignment (not a real FBA device).

If the ASSGN statement was entered from SYSLOG or if the LOG command was given, message 1T20I will be issued on SYSLOG to indicate which device has been selected. Restrictions:

- Do not use a generic assignment for a dummy device to be used as input or output device in a VSE/POWER-supported partition.
- If a configuration consists of mixed device types of the same device-class, such as 3375s and 3380s, then use either the actual device-type or an address list.

If a configuration includes FBA and CKD disk devices, specification of ANYDISK will assign any disk device to the logical unit SYSxxx. The parameters CKD and FBA (and others) permit more detailed specification of the disk device to be selected.

The specific device-type codes to which each device class applies are listed in Table 4.

Table 4. Device Class Assignments

READER	2540R, 3505, 1442N1, 2520B1, 3525RP
PRINTER	1403, 1403U, PRT1(3211), <3800, 3800B, 3800C, 3800BC>, PRT1(4248)
PUNCH	<2520B2, 2520B3>, 2540P, 1442N2, 3525P, 1442N1, 2520B1, 3525RP
TAPE	3420T9
CARTRIDGE	3480, 3490
TPA	TPAT128, TPAT256, TPAT384, TPAT512
DISK	3380, ECKD, FBA
CKD	3380
ANYFBA	<FBA, FBAV>
ANYDISK	<FBA, FBAV>, 3380, ECKD
DISKETTE	3540

Note: Devices are searched in the indicated order within their device class, except for devices enclosed within < >, which are searched simultaneously.

### device\_type

Use this specification if you are interested only in the specific type of device, and not in the physical unit. You can specify any disk, tape, printer, card punch or card reader device-type code shown in Table 1 on page 40. The system searches for the first free unit of the specified device-type. When a free unit is found, it is assigned to SYSxxx (see Table 5 on page 66). For disks, if SHR is specified, the first unit of the specified device-type is assigned, even if previously assigned.

If the ASSGN statement was entered from SYSLOG or if the LOG command was given, message 1T20I will be issued on SYSLOG to indicate which device has been selected.

Restriction: Do not use this specification for a dummy device to be used as input or output device in a VSE/POWER supported partition.

If you have a mixture of IBM 3380 models installed, an assignment such as  
 // ASSGN SYS009,3380,...

causes the system to select the first available IBM 3380 disk volume to be assigned, which may be any of the attached models. Therefore, if you want to assign a specific 3380 model, specify DISK (not 3380) and the required volume serial number in the VOL operand.

When you assign a 4248 printer using the device-type 4248, note that only those units added with the device code 4248 at IPL are available. A 4248 added as PRT1 will not be selected.

For a 3800 printing subsystem, you can use assignment by device codes as follows:

Specified code	is valid for			
	3800	3800B	3800C	3800BC
3800	X	X	X	X*,**
3800B		X		X*
3800C			X	X**
3800BC				X

\* The job cannot use the additional character generation storage feature.

\*\* The job cannot use the Burster-Trimmed-Stacker feature.

Specification of the device class PRINTER may select a 3800 from a list of printers; however, the existence of the two optional hardware features (the Burster-Trimmed-Stacker and additional character generation storage) cannot be assumed.

The three types of multiple device specification, device-list (cuu,cuu,...), or (cuu1:cuu2), device-class and device-type, assign the first available unit which matches the specification. However, the search order is different in each case:

**device-list** searches the units in the order specified in the list.

**device-class** searches the units of the specified class in the order of device-type codes as listed in Table 4 on page 64. For example, for device-class DISK, first all devices of device type 3380 are searched in ascending order of their channel numbers, then all devices of device type ECKD. For device-type groups enclosed within < >, the search is done simultaneously.

**device-type** searches the units of the specified type in ascending order of physical unit address.

Table 5 on page 66 shows an example of how the system scans the attached physical units with three different types of disk specification in the ASSGN statement/command.

Table 5. Device Search Order

IBM Device		Search Order		
Physical Unit	Device-Type Code	(280,281,183,382)	Disk	3380
181	3380		1	1
182	3380		2	2
183	3380	3	3	3
280	ECKD	1	6	
281	ECKD	2	7	
381	3380		4	4
382	3380	4	5	5

**mode**

Specifies mode settings for magnetic tapes (see Table 6). If the tape on the specified unit is at load point, the new mode setting becomes effective at once. For the 3480, 3490, 3490E, and 3590 tape units, the new mode setting always becomes effective at once, even if the tape is not at load point. For all other devices, the new mode becomes effective the next time load point is reached. If **mode** is not specified at IPL time, the default mode as shown in Table 6 applies.

For 800 bpi single-density 9-track tapes, a specification of C8 reduces the time required to OPEN an output file.

With dual-density tape units, the mode setting of D0 (6250 bpi) does not take effect for input tapes that were written at 1600 bpi (mode setting C0).

The standard mode is set during IPL. If the mode setting (different from, or the same as the standard mode) is specified in a temporary ASSGN statement, it becomes the current mode setting. This mode stays in effect until a subsequent assignment with a new mode or until EOJ. When the current job ends, the standard mode is restored, provided the unit was not unassigned during the job. The mode specification in a permanent ASSGN becomes the standard mode. If **mode** is not specified for a new job, the mode is the same as the standard mode or the mode specified in the last permanent assignment.

Table 6. Mode Settings for Tapes

Device Type	Mode	Default Mode	Density (bpi)	Characteristics		
				Parity	Convert Feature	Translate
3420T7		90				
	10		200	odd	on	off
	30		200	odd	off	off
	38		200	odd	off	on
	20		200	even	off	off
	28		200	even	off	on
	50		556	odd	on	off
	70		556	odd	off	off
	78		556	odd	off	on
	60		556	even	off	off
	68		556	even	off	on
	90		800	odd	on	off

Table 6. Mode Settings for Tapes (continued)

Device Type	Mode	Default Mode	Density (bpi)	Characteristics		
	B0		800	odd	off	off
	B8		800	odd	off	on
	A0		800	even	off	off
	A8		800	even	off	on
3420T9	C8	D0	800	Single/dual density 9-track tapes		
	C0		1600	Single/dual density 9-track tapes		
	D0		6250	Single/dual density 9-track tapes		
3480	00	3480: 00	n.a.	Buffered write mode		
3490	20	3490: 00		Unbuffered write mode		
3490E	08	3490E: 08		Improved data recording, buffered		
	28			Improved data recording, unbuffered		
TPA		08	n.a.	see end of this section for 3590 mode settings		

**ALT**

Indicates an alternate magnetic tape unit that is used when the capacity of the original assignment is reached.

This operand can only be specified for programs using logical IOCS. The specifications for the alternate unit are the same as those of the original unit. The characteristics of the alternate unit must be the same as those of the original unit. The original assignment and an alternate assignment must both be permanent or both be temporary assignments. Multiple alternates can be assigned to one symbolic unit.

When an alternate assignment becomes active, the system sets the mode for the alternate unit to that of the original unit. Therefore, the system accepts an alternate assignment only if the mode setting specified (explicitly or by default) for the original unit is also valid for the alternate unit.

Using multivolume tape files without specifying ALT mode can cause performance degradation, because the first tape has to be rewound and unloaded before the next tape can be mounted (except for 34xx devices with automatic cartridge loader, where the cartridges can be stacked).

If the original unit is reassigned, the alternate unit must also be reassigned. The ALT operand is invalid for SYSRDR, SYSIPT, SYSIN, SYSLNK, and SYSLOG.

**PERM | TEMP**

Indicates whether the assignment should be permanent (PERM) or temporary (TEMP). It is thus possible to override the // specification or omission.

A permanent assignment overrides the current assignment and deletes the stored permanent and all alternate assignments.

For dynamic partitions the duration of a permanent assignment corresponds to the lifetime of the dynamic partition.

This operand must be entered at the position shown in the syntax description.

**VOL=volser**

Specifies the volume serial number of the device required. This option may be specified only for tapes, disks, and diskettes.

If VOL is specified, the system searches for the first unit in the requested sequence and, if the unit is ready (for a tape, if it is at load point and not already assigned) checks the volume label to see if the required volume is mounted. If not, the next unit is checked, and so on until the proper volume serial number is found or until the end of the specified sequence is reached. The requested volume must be mounted on the unit specified in the message **1T50A MOUNT volser ON X'cuu'**.

**Note:** If, while reading the volume label, the job is canceled (for example, because of I/O errors), the device has to be reset to its initial status (using DVCDN and DVCUP commands) before the unit can be reassigned with a generic assignment.

If a volume serial number specified for a disk device does not match the actual volume serial number, the system notifies the operator and allows him to correct the assignment statement.

**Note:** In a mixed TPA device environment (3590, 3590E, 3590H, and 3592) a volume can be written with different track characteristics (128, 256, 384, and 512 tracks). Thus the specification of ASSGN SYSxxx,TPA,VOL=volser may cause the system to issue a request for a volume to be mounted on a device that cannot accommodate the specified volume. To make sure that the system selects a TPA device with the appropriate track characteristic specify the following:

- TPAT128 for a plain 3590 device
- TPAT256 for a 3590E device
- TPAT384 for a 3590H device
- TPAT512 for a 3592 device

Also the parameters cuu or address-list can be used in a mixed TPA environment.

**SHR**

This option can be specified only for disk devices and is meaningful only in combination with **address-list**, **device-class**, and **device-type** (see the descriptions of these operands.) It means that the unit can be assigned to a disk device which is already assigned. If the option is not specified, the system assigns the unit to a disk device not yet assigned. Therefore, unless a private device is required, it is recommended to use the SHR operand in combination with generic assignments.

**IBM 3590 Mode Settings**

The mode settings are defined as follows:

Bit 0	Reserved
Bit 1	FIFO Read Buffer
Bit 2	Inhibit Buffered Write
Bit 3	Inhibit Supervisor Commands
Bit 4	Data Compaction
Bit 5 - 7	Write Format 0 - 7

VSE allows the following mode settings to be selected by the operator:

- X'00' .. X'0F'
- X'20' .. X'2F'

Bit 3 of the mode setting byte (Inhibit Supervisor Commands) will always be set internally by the I/O Supervisor.

If a mode setting is specified that is not supported by the device, the first I/O to the device is rejected with message 0P18I.

Although IBM 3590 currently only accepts write format 0 and 7, format IDs 0 to 7 are supported by z/VSE, because they are part of the Tape Products Architecture, and may be used by future models. The write format IDs are interpreted as follows:

- 0 Device default format
- 1-6 Mode-dependent formats
- 7 Format currently used on the medium

The VSE device default mode is X'08'.

## BANDID

### BANDID (Mount or Query 4248 Print Band)

This command only applies to IBM 4248 printers. It allows you to find out which print band is mounted, or to specify which band is to be mounted.

#### AR Format

►►—BANDID *cuu*—┬──┬──┬──┬──►  
                  └─,band\_id─┘ └─,FOLD─┘ └─,NOCHK─┘

#### **cuu**

Specifies the device number of the printer for which you issue the command.

#### **band-id**

Specifies the identifier of the print band that is to be mounted.

If you omit the operand, then you get a message indicating the identifier of the currently mounted band.

#### **FOLD**

Causes lowercase characters to be printed as uppercase characters.

#### **NOCHK**

Causes a data check to be suppressed if it resulted from a mismatch between a print character and the band-image buffer.



## BATCH (Start or Continue Processing)

The BATCH command activates or continues processing in one of the foreground partitions or continues processing in the background partition. The BATCH command is not allowed in a dynamic partition.

The function of the BATCH command is exactly the same as that of the START command. If the specified partition is available, job control reads the operator's next command from SYSLOG. When the operator desires to give control to another command input device, he makes an assignment to SYSRDR or SYSIN, and presses the END or ENTER key.

If the specified partition has been made inactive by an UNBATCH command, it is made active. If the partition was temporarily halted by a STOP command, it is restarted. If the partition is in operation, it continues, and message

1P1nD AREA NOT AVAILABLE OR PARTITION ACTIVE

is issued to the operator. In either instance, attention routine communication with the operator terminates following the BATCH command.

### AR Format



#### BG

Indicates that the background partition is to be reactivated.

**Fn** Indicates that the specified foreground partition is to be activated or restarted after having been stopped by a STOP command.

If the operand is omitted, BG is assumed.

## CACHE (Control Cache Operations)

The CACHE command can only be used in conjunction with an IBM 3990-3 or 3990-6 DASD Storage Controller. It requires cache storage to be installed and enables the VSE system operator to

- Make an addressed device (actuator) eligible or not eligible for caching operations.
- Make subsystem storage or functions available or not available for caching operations.

VSE supports **Basic Caching**, which is basically a read caching.

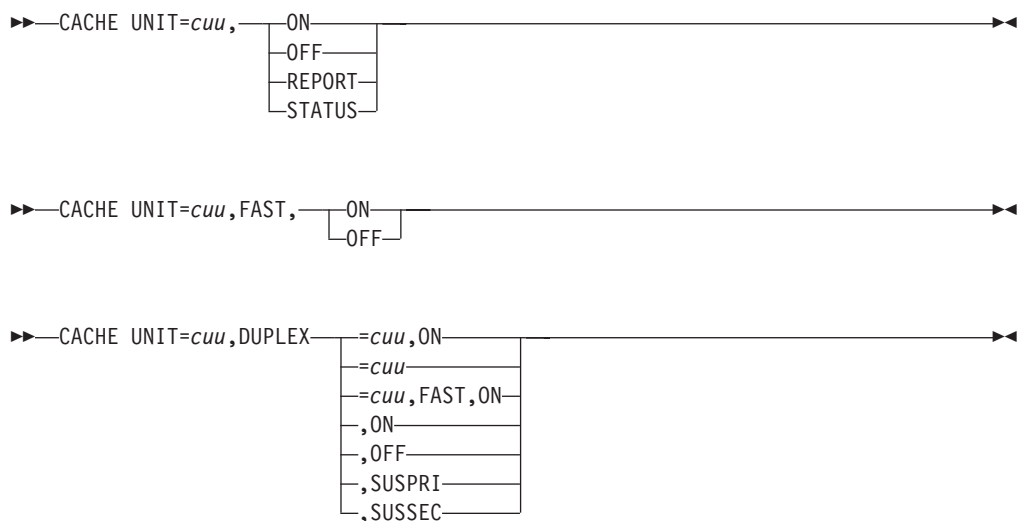
VSE also supports the following 3990-3 / 3990-6 extended cache functions:

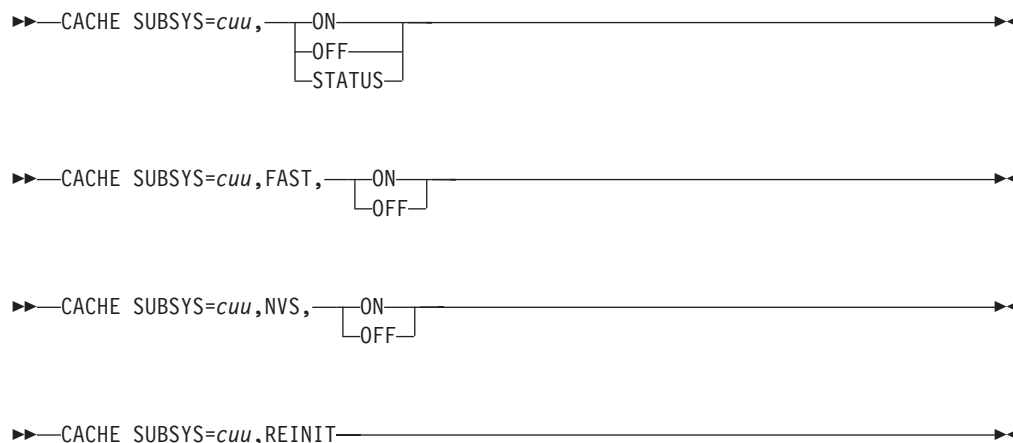
- **DASD Fast Write**, which uses the NVS non-volatile storage to provide fast write access, before writing data to DASD.
- **Cache Fast Write**, which provides fast write access for data not necessarily required to reside on permanent DASD.
- **Dual Copy**, which allows to control the creation, modification, and resetting of duplex pairs and to display the actual status of the DASD devices.

The 3990-3 / 3990-6 dual copy function allows you to maintain logically identical copies of a DASD device on two different volumes in the same subsystem. To use dual copy, you establish two devices as a **duplex pair**. One device is designated as the **primary** and the other as the **secondary** device. The primary is online. You must set the secondary device down (DVCDN) in **all** systems before you establish a duplex pair. When the duplex pair has been reset to simplex, the secondary device must be set up again (DVCUP) before it can be accessed.

For better control of these caching functions, caching is formally composed of device caching and subsystem caching. Both must be set to accomplish that any caching is in effect for a specific device.

The CACHE command must be given in one or more of the following (AR) formats:





**UNIT=*cuu***

Specifies the device to which cached access is either allowed (**ON**) or disallowed (**OFF**).

**SUBSYS=*cuu***

Specifies the subsystem cache or storage to which access is either allowed (**ON**) or disallowed (**OFF**). For *cuu* you can specify any device within the subsystem. The function you request affects **all** devices in the subsystem.

**ON**

Must be used to activate any of the functions described above.

**OFF**

Must be used to deactivate any of the functions described above.

**FAST**

Specifies that fast write access is either to be allowed (**ON**) or disallowed (**OFF**) for

- The addressed device (DASD Fast Write for **UNIT=*cuu***), or
- All devices in the subsystem (Cache Fast Write for **SUBSYS=*cuu***).

**NVS**

Specifies that non-volatile storage is to be made available or unavailable to the subsystem.

**REPORT**

Provides statistical performance data for the addressed device. Note that counters are reset at control unit IML.

Sample output:

CACHE UNIT=*cuu*,REPORT may produce the following output:

# CACHE

REQUESTS:	READ		WRITE		
	TOTAL	CACHE-RD (hits)	TOTAL	CACHE-WRT (hits)	DASD FAST-WRT (all)
NORMAL	A1	B1	C1	D1	E1
SEQUENTIAL	A2	B2	C2	D2	E2
CACHE FAST WRITE	A3	B3	C3	D3	N/A
TOTALS	A	B	C	D	E

REQUESTS: (read and write)	
INH. CACHE LOAD	F1
BYPASS CACHE	F2

DATA TRANSFERS:	DASD->CACHE	CACHE->DASD
NORMAL	G1	H1
SEQUENTIAL	G2	N/A

Figure 4. Cache Statistics for 3990-3 / 3990-6

Note that each line shows different cache settings in a channel program. Channel programs where record caching is used may not be included in the totals.

### CACHE-RD

Indicates all read-I/Os which did not require any data movement from DASD (read hits).

### WRITE TOTAL

Includes DASD fast write and non-DASD fast write requests (for example, CKD channel programs, without DEFINE EXTENT).

### CACHE-WRT

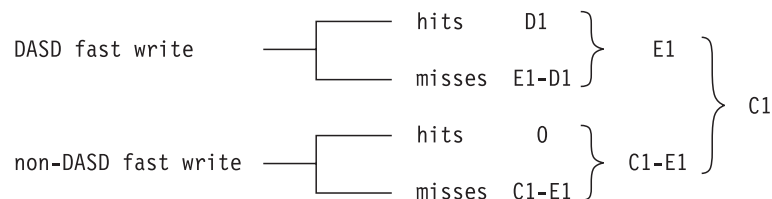
Includes all I/Os (with at least one write command) that - as far as performance is concerned - profited from the cache (write hits, D1 and D2 via DASD fast write, D3 via cache fast write).

### DASD FAST-WRT

Includes all requests (hits and misses).

### NORMAL WRITE REQUEST

For normal write requests (for example, not sequential, no CFW) the counter holds:



Write hits only exist for DASD fast write. If all writes are DASD fast writes, C1 and E1 counters are identical.

Calculable hit ratios:

$$\text{Read } B/A \quad \text{DASD fast write } (D1+D2)/E \quad \text{Cache fast write } D3/C3$$

Cache fast write and DASD fast write are exclusive; therefore, 'not applicable' is shown in the CACHE FAST WRITE line.

### INHIBIT CACHE and BYPASS CACHE

Include reads and writes. They are not contained in the A or C counters.

**DATA TRANSFERS**

The counters G are the number of transfers to stage the cache from DASD. Those tracks being read ahead via sequential access caching are counted separately.

Counter H1 designates all cache to DASD de-staging transfers (write from cache to physical device).

**STATUS**

Provides overall caching information for the addressed device or subsystem. For the dual copy function, STATUS lists the actual dual copy status of a device.

Example:

CACHE SUBSYS=130,STATUS may produce the following output:

```

SUBSYSTEM CACHING STATUS: ACTIVE
  CACHE FAST WRITE: ACTIVE
    CACHE STORAGE: CONFIG.      .....K
                      AVAIL.      .....K
                      PINNED      .....K
                      OFFLINED    .....K
      NVS STATUS: AVAILABLE
    NVS STORAGE: CONFIG.      .....K
                      PINNED      .....K
    
```

CACHE UNIT=130,STATUS:

```

DEVICE CACHING STATUS: ACTIVE
  DASD FAST WRITE: ACTIVE
    DUAL COPY STATUS: SIMPLEX
      PRIMARY DEVICE: ....
      SECONDARY DEVICE: ....
      PINNED DATA FOR: CYL=..... TRK=..
    
```

**DUPLEX=cuu,ON**

Establishes dual copy for the primary device UNIT and the secondary device DUPLEX from simplex state. The secondary device must be down (with the DVCDN command). Paired devices must be attached to the same logical DASD subsystem and must meet the 3990 compatibility requirements. The entire primary DASD is copied to the secondary device.

**Note:** The establishment of a duplex pair from simplex status takes several minutes; during this phase the duplex pair status remains PENDING DUPLEX.

**DUPLEX=cuu**

This command is the same as DUPLEX=cuu,ON except that no copy is taken (for example, in cases where both DASDs are initialized).

**DUPLEX=cuu,FAST,ON**

This command is the same as DUPLEX=cuu,ON except that the copy is taken at maximum speed. While this command is in progress, the primary device returns 'busy' to all other accesses. It is recommended only for those cases where a device-busy status extending over a few minutes has no impact on other tasks in the system.

**DUPLEX,ON**

If a duplex pair is in suspended state, it can be set to duplex by specifying the primary device (UNIT=), DUPLEX, and ON. Only those tracks that were modified since entering the suspended state are copied.

## CACHE

If a duplex pair is in suspended state and the secondary device has to be replaced by a new DASD, the suspended primary device and the new secondary device can be set to duplex by specifying the suspended primary device (UNIT=cuu), a new secondary device (DUPLEX=cuu), and ON. The entire primary device is copied to the new secondary device.

### DUPLEX,OFF

Switches dual copy off for the duplex pair with primary device UNIT=cuu. When the devices are changed from a duplex pair to simplex devices, the old primary device retains the DASD fast write and the device caching status of the duplex pair.

### DUPLEX,SUSPRI

Suspends a duplex pair with primary device UNIT and failing primary device. The subsystem swaps the primary and secondary devices in this case, because the suspended device is always the secondary device.

### DUPLEX,SUSSEC

Suspends a duplex pair with primary device UNIT and failing secondary device.

### REINIT

Resets the 3990 controller to its default values. The command terminates all duplex pairs. Data in cache and NVS is lost. The default values after REINIT are:

- Subsystem caching available.
- Device caching active.
- NVS not available.
- Cache Fast Write active.
- DASD Fast Write inactive.
- Dual Copy disabled.

CACHE SUBSYS=cuu,REINIT is rejected by the control unit if the specified cuu is a secondary device of a duplex pair.

## Summary

The following table shows the type of commands required for the setting and querying of the individual cache functions:

Cache Function	Setting	Status Information via
Basic Caching (a)	UNIT SUBSYS=cuu,ON OFF	UNIT SUBSYS=cuu,STATUS
DASD Fast Write (b)	UNIT=cuu,FAST,ON OFF	UNIT=cuu,STATUS
NVS (b)	SUBSYS=cuu,NVS,ON OFF	SUBSYS=cuu,STATUS
Cache Fast Write (c)	SUBSYS=cuu,FAST,ON OFF	SUBSYS=cuu,STATUS

- (a) Basic caching is only in effect if it is set on UNIT and on SUBSYS level, in any sequence.
- (b) DASD Fast Write can only be in effect if NVS is set on, too.
- (c) Setting DASD or Cache Fast Write on also requires that subsystem caching is on.

Note that the settings with CACHE UNIT=... remain in effect even across control unit or device power off or IMLs.

## Examples

### CACHE UNIT=cuu,ON

Activates device caching for the specified device.

**CACHE UNIT=cuu,OFF**

Deactivates device caching for the specified device.

**CACHE SUBSYS=cuu,ON**

Activates subsystem caching for the entire specified subsystem.

**CACHE UNIT=cuu,FAST,ON**

Enables DASD fast write access for the specified device.

**CACHE SUBSYS=cuu,NVS,ON**

Makes non-volatile storage available for the subsystem.

**Example for Activation of Dual Copy:**

1. First check subsystem cache settings: **CACHE SUBSYS=cuu,STATUS**
2. If basic caching and NVS are switched on, go to 5, else continue.
3. Switch subsystem caching on: **CACHE SUBSYS=cuu,ON**
4. Or switch NVS on: **CACHE SUBSYS=cuu,NVS,ON**
5. Check device caching for both devices: **CACHE UNIT=cuu,STATUS**
6. If device caching is off, go to 8, else continue.
7. Set device caching off: **CACHE UNIT=cuu,OFF**
8. Set the secondary device down: **DVCDN cuu** (job control)
9. Establish duplex pair: **CACHE UNIT=cuu,DUPLEX=cuu,ON**

**Operator Responses**

The following operator responses are given as *subsystem status*:

**CACHE STATUS: status**

where:

**status**

is the status of the cache, which can be any of the following:

**ACTIVE** if the cache subsystem is active.

**CACHE ON PENDING**  
if the cache is being brought online.

**FORCED OFFLINE**  
when an internal subsystem error caused caching termination.

**DEACTIVATED**  
when caching termination was forced by a user command.

**CACHE OFF PENDING**  
when a deactivation operation is in progress.

**CACHE OFF FAILURE**  
when a deactivation operation failed.

**CACHE STORAGE: status bytes(K)**

where:

**status**

can be any of the following:

**CONFIGURED**  
configured cache capacity.

**AVAILABLE** number of bytes of cache available to this subsystem for cache space.

## CACHE

**PINNED**            number of bytes of pinned data in cache.  
**OFFLINED**        number of bytes of cache unavailable to the storage director because of cache read failures.

**bytes**  
indicates the quantity of cache storage (in bytes or K-bytes) which is currently in the specified status.

### CACHE FAST WRITE: status

where:

#### status

can be any of the following:

**ACTIVE**            if cache fast write is active.

**DEACTIVATED**  
when cache fast write is disabled.

### NVS STATUS: status

where:

#### status

can be any of the following:

**AVAILABLE**        if NVS is active.

**FORCED UNAVAILABLE**  
when an internal subsystem error caused NVS termination.

**UNAVAILABLE**  
when NVS termination was forced by a user command.

**NVS OFF PENDING**  
when a de-stage operation is in progress.

### NVS STORAGE: status bytes(K)

where:

#### status

can be any of the following:

**CONFIGURED**  
configured NVS capacity.

**PINNED**            number of bytes of pinned data in NVS.

**bytes**  
indicates the quantity of NVS (in bytes or K-bytes) which is currently in the specified status.

The following operator responses are given as *device status*:

### DEVICE CACHING STATUS: status

where:

#### status

can be any of the following

**ACTIVE**            if caching for device is active.

**CACHE OFF FAILURE**  
when transfer of modified data to DASD has failed.

**DEACTIVATED**  
when caching for device is disabled.



**DASD FAST WRITE: status**

where:

**status**

can be any of the following

**ACTIVE** if DASD fast write is active.

**CACHE OFF FAILURE**

when transfer of modified DASD fast write data to DASD failed.

**DEACTIVATED**

when DASD fast write is disabled.

**DUAL COPY STATUS: status**

**PRIMARY DEVICE: cuu**

**SECONDARY DEVICE: cuu**

where:

**status**

can be any of the following

**SIMPLEX** if device is in simplex mode.

**DUPLEX** if the duplex pair is active.

**PENDING DUPLEX**

when the copy to establish a duplex pair is in progress.

**SUSPENDED PRIMARY**

if the primary of a duplex pair is suspended by a host command or by the subsystem.

**SUSPENDED SECONDARY**

if the secondary of a duplex pair is suspended by a host command or by the subsystem.

**cuu**

is the device number of the device on which the I/O operation occurred and the other device of the duplex pair.

**PINNED DATA FOR: CYL=..... TRK=..**

where:

**CYL** is the cylinder for which pinned data exists.

**TRK** is the track for which pinned data exists.

## CANCEL

### CANCEL (Cancel Job or I/O Request)

The CANCEL command, when used as a job control command (JCC), cancels the execution of the current job in the partition in which the command is given. No dump is produced by the CANCEL job control command. If a dump is required, use the attention routine command.

When issued as an attention routine (AR) command, CANCEL may be used for the following purposes:

- To cancel an I/O request on a device for which operator intervention was requested.
- To cancel the execution of the current job in the specified partition and, optionally, to override the dump options existing for that partition.
- To cancel the command that is currently processed by the Attention Routine, regardless of the console that issued the command.

The AR CANCEL command is accepted only when the attention routine is available. If the attention routine is not available when you want to enter the AR command, enter the RC command (Request Communication), and enter the CANCEL command in response to the message 1I40I READY.

#### JCC Format

▶▶—CANCEL—▶▶

#### AR Format

▶▶—CANCEL *cuu*—▶▶

#### AR Format

▶▶—CANCEL — BG —▶▶  
          |Fn| —▶▶  
          |*dyn\_partition*| —▶▶  
          |*jobname*| —▶▶  
          |,DUMP| —▶▶  
          |,PARTDUMP| —▶▶  
          |,NODUMP| —▶▶  
          |,SYSDUMP| —▶▶  
          |,NOSYSDUMP| —▶▶  
          |,FORCE| —▶▶  
          |,SKIPAB| —▶▶

#### AR Format

▶▶—CANCEL AR—▶▶

#### **cuu**

Indicates that the I/O request for the specified device is to be canceled. Note that *cuu* must be a *three-digit* device number, which can be any value between 000 and FFF. (It is the number by which the device was defined during I/O configuration.)

#### **BG | Fn | dyn\_partition**

Indicates that the job in the specified (static or dynamic) partition is to be canceled.

#### **jobname**

Indicates the job name of the job to be canceled. *jobname* can be up to 8 characters and must be unique.

**DUMP**

Causes a dump of the registers, of the supervisor, of the partition, the used part of the system GETVIS area, and of the SVA phase in error (if the error occurred in the SVA).

**PARTDUMP**

Causes a dump of the registers, of supervisor control blocks, of the partition, of areas acquired through GETVIS in the partition, and of the SVA phase in error (if the error occurred in the SVA).

**NODUMP**

Suppresses the DUMP option.

**SYSDUMP**

Indicates that dumps are to be written to the dump sublibrary which is defined for the appropriate partition. If no LIBDEF DUMP statement is in effect for the partition in question, or if the defined sublibrary is full, the system assumes the option NOSYSDUMP. The form SYSDMP is accepted for compatibility reasons.

**NOSYSDUMP**

Indicates that dumps are to be written on SYSLST. The form NOSYSDMP is accepted for compatibility reasons.

**FORCE**

Causes the Cancel command to be carried out immediately, even if a critical system function has requested a delay. Any action specified in an ON \$CANCEL statement is **not** carried out.

**Note:** Use the FORCE operand **with caution**, and only when a Cancel command without FORCE has failed to terminate the job.

The use of this operand can cause critical system functions to be interrupted. This, in turn, can lead to inconsistencies in the system (for example, library directories not updated).

**SKIPAB**

The option is mutually exclusive to option FORCE. It causes abnormal termination exit processing to be skipped. All other system functions executed during a normal cancel process are performed.

**AR**

Causes the command that is currently processed by the Attention Routine to be terminated abnormally, regardless of the console that issued it. This command requires master authorization.

If the CANCEL command is issued for a partition in which the subsystem VSE/POWER is active, a message is issued to the operator to verify the request for cancelling that partition.

The remaining statements and data will be skipped up to /& or to the label specified in an ON \$CANCEL GOTO statement for the job. (If FORCE is specified, the ON \$CANCEL statement is not carried out.) Current<sup>®</sup> exception: If the JOB statement was omitted, and CANCEL was a statement in a procedure, then all procedure processing is terminated and **the next statement on SYSRDR** is executed.

# CLOSE

## CLOSE (Close Output Logical Unit)

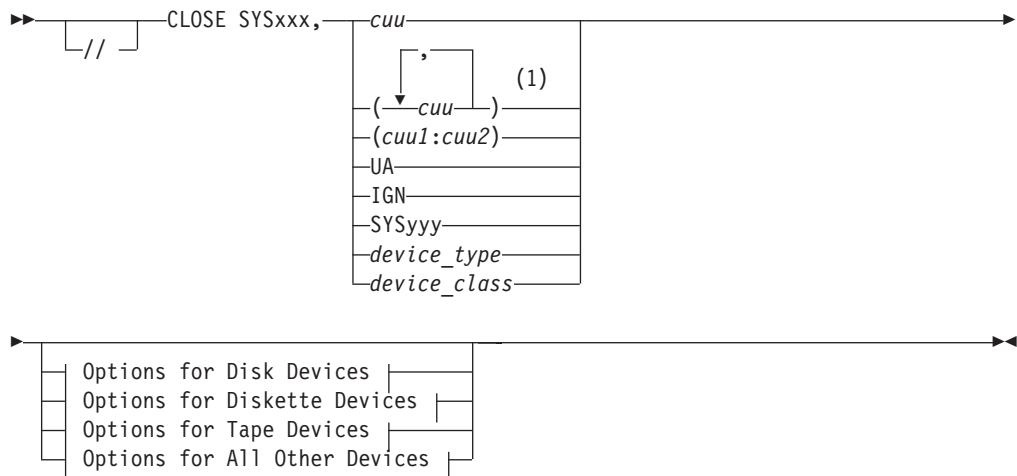
The **CLOSE command** is used to close either a system or programmer logical unit assigned to a tape, or a system logical unit assigned to a disk or diskette.

The **CLOSE statement** is used to close either a system or programmer logical unit assigned to tape. It only applies to temporarily assigned logical units.

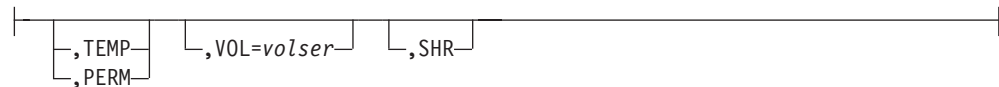
The logical unit can optionally be reassigned to another device, unassigned, or, in the case of a magnetic tape file, switched to an alternate unit. When SYSxxx is a system logical unit (SYSLST, SYSPCH, etc.), one of the optional parameters **must** be specified. When closing a programmer logical unit (SYS000-SYS254), no optional parameter need be specified. When none is specified, the programmer logical unit is closed and the assignment remains unchanged.

Closing a magnetic tape unit causes the system to write a tapemark, an EOV trailer record, and two tapemarks, and to rewind and unload the tape. The trailer record contains no block count, and later access by logical IOCS may result in a 4131D message, which can be ignored.

### JCC, JCS Format

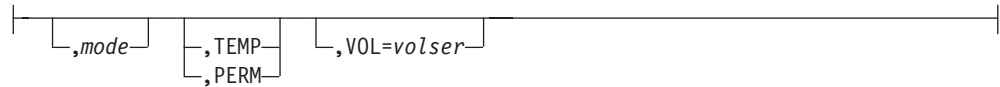


#### Options for Disk Devices:



#### Options for Diskette Devices:



**Options for Tape Devices:****Options for All Other Devices:****Notes:**

1 Up to 64 device addresses can be specified.

**Note:** When SYSxxx is a system logical unit (SYSLST, SYSPCH, etc.), one of the optional parameters **must** be specified.

**SYSxxx**

For the CLOSE command only: For disk or diskette: SYSIN, SYSRDR, SYSIPT, SYSPCH, or SYSLST.

For both the statement and the command: For magnetic tape: SYSPCH, SYSLST, SYSOUT, or SYS000-SYS254.

**cuu**

Specifies that, after the logical unit is closed, it will be assigned to the channel and unit. **c** is the channel number, **uu** is the unit number, in hexadecimal. In the case of a system logical unit, the new unit will be opened if it is either a disk, diskette, or a magnetic tape at load point.

**mode**

Device specification for mode settings on 7-track and 9-track tape. The specifications are shown in Table 6 on page 66. If **mode** is not specified, the mode settings remain unchanged. The LISTIO command may be used to determine the current mode settings for all magnetic tape units.

**UA**

Specifies that the logical unit is to be (permanently) unassigned after the file has been closed.

**IGN**

Specifies that the logical unit is to be (permanently) unassigned after the associated file has been closed. Any subsequent references to the unit will be ignored until a new ASSGN is given for the unit, or IPL is performed. This operand is invalid for SYSRDR, SYSIPT, or SYSIN.

**ALT**

Specifies that the logical unit is to be closed and an alternate unit is to be opened and used. This operand is valid only for system output logical units (SYSPCH, SYSLST, or SYSOUT) currently assigned to a magnetic tape unit.

**SYSyyy**

Specifies that, after SYSxxx is closed, it will be assigned to the physical device to which SYSyyy is currently assigned (and to which it remains assigned). If SYSxxx is a system logical unit, it will be opened if the target device is a disk, diskette, or magnetic tape at load point, and if SYSxxx is not already assigned.

## CLOSE

### **device\_class**

Indicates that after the logical unit is closed, it will be assigned to the first available unit within the specified device class. The device classes and the device types to which they apply are listed in Table 4 on page 64.

### **device\_type**

Indicates that after the logical unit is closed, it will be assigned to the first free unit of the specified device type. The device-type codes which you can specify are shown in Table 1 on page 40.

### **TEMP**

Indicates that after the logical unit is closed, it will be temporarily assigned to the specified cuu.

### **PERM**

Indicates that after the logical unit is closed, it will be permanently assigned to the specified cuu.

### **VOL=volser**

Indicates that after the logical unit is closed, it will be assigned to the physical device with the specified volume serial number.

### **SHR**

Indicates that after the logical unit is closed, it can be assigned to a disk device which is already assigned.

## DATE (Override System Date)

The DATE statement places the specified date (including century information) temporarily in the communication region's job date field (JOBDATE). Utilities (for example LISTLOG), language translators (for example the High Level Assembler) and user applications may use this date for identifying printed output. This date is also displayed in the end-of-job message (see /& statement).

The date specified in the DATE statement applies only to the current job being executed. It is reset to the system date during end-of-job processing.

### JCS Format



### Notes:

- 1 The DATE option of the STDOPT statement indicates whether the first or the second format is actually in use

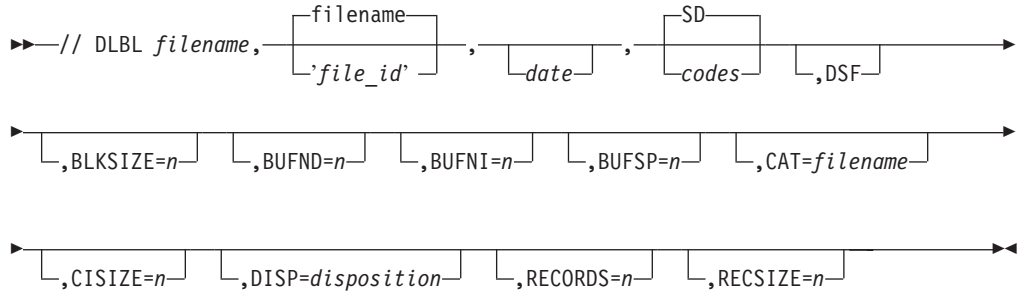
If a job or job step executes past midnight, the date given in the DATE statement is not incremented.

## DLBL (Disk Label Information)

The DLBL statement (disk label information) contains file label information for disk or diskette label checking and creation.

If OPTION USRLABEL is in effect, label information submitted for a job or job step is overwritten by any following job or job step.

### JCS Format



Continuation lines are accepted for the DLBL statement.

#### filename

This can be from one to seven alphanumeric characters, the first of which must be alphabetic, @, # or \$. This unique filename is identical to the symbolic name of the program DTF that identifies the file.

**Note:** Do not use the same filename for both a DLBL and a TLBL statement.

For VSE/VSAM, filename is identical to the **dname** of the **FILE (dname)** parameter and the **DDNAME=filename** parameter of the access method control block (ACB) in the processing program that identifies the file. If the DDNAME parameter is omitted, the filename must be contained in the symbolic name (label) field of the ACB.

#### 'file-id'

This is the unique name associated with the file on the volume. This can be from one to 44 characters, contained within quotes, including file-id and, if used, generation number and version number of generation. Within the identifier, a character sequence of a quote followed by a comma or a blank is not allowed. The system interprets this sequence as the end of the identifier.

If fewer than 44 characters are used, the field is left-justified and padded with blanks. If this operand is omitted, **filename** is used. The diskette uses a maximum of eight characters in file-id.

For VSE/VSAM, **file-id** must be specified when a file is being processed. The file-id is identical to the name of the file, specified in the DEFINE command and listed in the VSE/VSAM catalog. For VSE/VSAM, the file-id must be coded according to the following rules:

- One to 44 characters long, enclosed in quotes (');
- Characters must be alphanumeric (A-Z, 0-9, @, \$, or #) or hyphen (-) or plus zero (+0);
- After each group of eight or fewer characters, a period (.) must be inserted;
- No embedded blanks are allowed;



- The first character of the file-id and the first character following a period must be alphabetic (A-Z) or @, \$ or #.

For details on the VSE/VSAM partition/processor unique file-id (%%), see “VTOC Label Processing” in the the *VSE/VSAM User’s Guide and Application Programming*.

#### date

The DATE operand can be omitted, or can be supplied in one of two different formats as shown below:

- Retention period format. This is specified:
  - As a decimal number nnnn (0-9999)

or, equivalently,

- as 00/nnnn (0-9999)

The nnnn is one through four decimal digits and specifies the retention period in days. You may use retention periods for new files to help reduce the chance of later accidental deletion. After the retention period, the file can be deleted or written over by another file.

Internally the system converts the retention period into a expiration date by adding up the retention period and the creation date.

- Date format. This is specified as:
  - yy/ddd with yy not equal to 00
  - 19yy/ddd
  - 20yy/ddd

The yy is a two-digit year number (00 through 99) and the ddd is a three-digit day number from 000 through 366. You may use this date format to specify the expiration date for a new file. On and after the expiration date, the file can be deleted or written over by another file.

Files with an expiration date of 1999/366 are always considered unexpired. Files with an expiration date of 1999/365 are considered unexpired, with one exception: if the expiration date 1999/365 was caused by the specification of a retention period. For example, a file created on 11/30/1999 with a retention period of 31 will expire.

The format yy/ddd will be complemented by the system to either 19yy/ddd or 20yy/ddd, dependent on the current date’s year and yy. The system will complement yy/ddd to 19yy/ddd if 19yy is greater than or equal to the current date’s year, and to 20yy/ddd else. For example, in 1998, the expiration date 98/ddd is complemented to 1998/ddd, whereas 97/ddd is complemented to 2097/ddd. This is because expiration dates are considered to be future-oriented rather than past-oriented.

If this operand is omitted, a 7-day retention period is assumed. If this operand is present for an input file, it is ignored. For VSE/VSAM, this parameter overrides the expiration date specified in the DEFINE command. However, VSE/VSAM files or data spaces can only be deleted through the DELETE command even though the expiration date has been reached.

#### codes

This is a two to four character field indicating the type of file label, as follows:

<b>SD</b>	for sequential disk or for DTFPH with MOUNTED=SINGLE
<b>DA</b>	for direct access or for DTFPH with MOUNTED=ALL
<b>DU</b>	for diskette
<b>ISC</b>	for indexed sequential using load create

**ISE** for indexed sequential using load extension, add, or retrieve  
**VSAM** for all Virtual Storage Access Method files

If this operand is omitted, SD is assumed.

#### **DSF**

This operand indicates that a data-secured file is to be created or processed. At OPEN time, if a data-secured file is accessed, a warning message is issued to the operator, who then decides whether the file may be accessed.

For the diskette and for VSE/VSAM this operand is ignored by OPEN. All VSE/VSAM files are data secured. The DSF operand is not required for an **input** file, and it does not invoke the support if the file was not originally created as a data-secured file.

#### **BLKSIZE=n**

This operand permits specification of a block size different from that given in the DTFSD macro for sequential disk files. This allows the user to utilize a more effective blocking factor. The parameter is ignored for all DTF types except DTFSD. It is not valid for VSE/VSAM files, or files on FBA devices. The value specified for n must not exceed 65,535.

If the file contains blocked fixed-length records, n must be:

- For input files: a multiple of the RECSIZE value;
- For output Files: 8 + a multiple of the RECSIZE value. The additional 8 bytes allow for a count field for system use.

Note that this parameter will be accepted by Job Control, but the job will later be canceled if the value specified is not a multiple of the RECSIZE value. The job will also be canceled if the BLKSIZE operand in the DTFSD macro is specified for TYPEFLE=WORK. For further details on the DTFSD macro, see "DTFSD Macro" in *z/VSE System Macros Reference*.

#### **BUFND=n**

Specifies, for a VSE/VSAM file, the number of I/O buffers to hold control intervals containing data records. Each buffer is the size of one data control interval. This specification overrides the value given for BUFND in the ACB macro (provided the DLBL value is greater than the ACB value). See "I/O Buffer Space" in *VSE/VSAM User's Guide and Application Programming* for further details.

#### **BUFNI=n**

Specifies, for a VSE/VSAM file, the number of I/O buffers to hold control intervals containing index records. Each buffer is the size of one index control interval. This specification overrides the value given for BUFNI in the ACB macro (provided the DLBL value is greater than the ACB value). See *VSE/VSAM User's Guide and Application Programming* for further details.

#### **BUFSP=n**

If a VSE/VSAM file is to be processed, this operand specifies the number of bytes of virtual storage (0-9999999) to be allocated as buffer space for this file. It overrides the values specified for BUFSP in the ACB macro and for BUFFERSPACE in the DEFINE command (only if the value of the DLBL BUFSP operand is greater than the value of the BUFFERSPACE parameter). See "I/O Buffer Space" in *VSE/VSAM User's Guide and Application Programming* for further details.

#### **CAT=filename**

This operand is valid in a DLBL statement for a VSE/VSAM file only. It

specifies the filename (1 to 7 alphanumeric characters) of the DLBL statement for the catalog owning this VSE/VSAM file. The system searches only this catalog for the file-id when the VSE/VSAM file is to be opened. Specify this operand only if you want to override the system's assumption that the job catalog or, if there is no job catalog, that the master catalog owns the file.

In a system **with** a job catalog specify nothing for the job catalog, a private name for a private user catalog, or IJSYSCT for the master catalog.

In a system **without** a job catalog specify nothing for the master catalog, or a private name for a private user catalog.

The only Access Method Services commands that use the CAT operand to specify a private user catalog are the PRINT, REPRO, VERIFY, and DELETE ERASE commands.

#### **CISIZE=n**

This operand permits specification of a control interval size for SAM files on FBA devices. The size overrides that specified (or defaulted) in the respective DTF macro. The specified size must be a number from 512 to 32,768 and a multiple of the FBA block size; if it is greater than 8K, it must be a multiple of 2K.

This operand is valid only for DLBL statements with the code SD.

#### **DISP=disposition**

This operand is valid only in a DLBL statement for a VSE/VSAM file. It permits specification of the data set disposition. The three positional keywords tell VSAM how the file is to be:

- Opened (keyword1);
- Closed after normal termination of the job step (keyword2);
- Closed after abnormal termination or cancelation of the job step (keyword3).

**disposition** can be specified in one of the following formats:

```
keyword1
(keyword1,keyword2)
(keyword1,keyword2,keyword3)
```

where:

```
keyword1 may be NEW or OLD
keyword2 may be DELETE, KEEP or DATE
keyword3 may be DELETE or KEEP
```

If you use the parenthesis syntax, each keyword (but not the separating commas) may be omitted. For example, the following three specifications are equivalent:

- DISP=NEW
- DISP=(NEW,)
- DISP=(NEW,,)

Specifying DISP=(,) or DISP=(,,) is the same as if the whole DISP keyword had been omitted.

For the meaning of these keywords, and the default values, see "Format of the DLBL Statement" in *VSE/VSAM User's Guide and Application Programming* .

#### **RECORDS=n**

This operand is only valid for VSE/VSAM space management for SAM feature

## DLBL

files. It permits specification of the number of records for the primary and secondary data set allocation. The operand can be specified in one of two formats:

RECORDS=*n*  
RECORDS=(*n*,*n1*)

where *n* indicates the number of records for the primary allocation, and *n1* the number of records for the secondary allocation. *n* must not be zero; *n1* may be larger or smaller than *n*.

The RECORDS and RECSIZE operands must either both be specified or both be omitted. For further details, see "Format of the DLBL Statement" in *VSE/VSAM User's Guide and Application Programming*.

### **RECSIZE=*n***

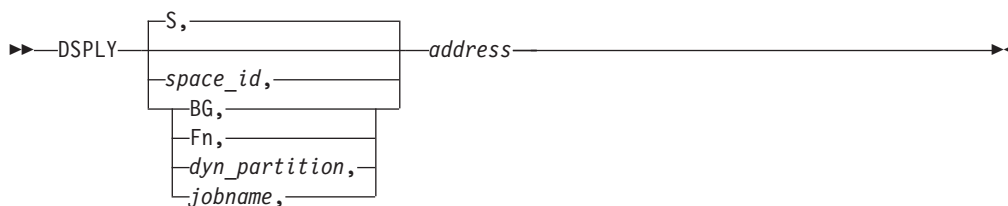
This operand is valid only for VSE/VSAM space management for SAM feature files. It permits specification of the average record length of the file. The value specified for *n* must not be zero. The RECSIZE and RECORDS operands must either both be specified or both be omitted. For further details, see "Format of the DLBL Statement" in *VSE/VSAM User's Guide and Application Programming*.

## DSPLY (Display Virtual Storage)

The DSPLY command allows the operator to display 16 bytes of virtual storage, starting at the specified hexadecimal address, on the device assigned to SYSLOG. Two characters (0-9, A-F) appear on SYSLOG for each byte of information; these characters represent the hexadecimal equivalent of the current information in virtual storage. In addition, an EBCDIC translation of the displayed storage is shown.

**Note:** In a multiprocessor environment the output of the DSPLY command for addresses lower than x'1000' is random. It shows storage particular to one of the active CPUs.

### AR Format



#### space\_id

Indicates in which address space the specified address is to be displayed. Valid specifications are:

R (real) or S (shared)  
0 through 9, A, B

The default value is S.

#### BG | Fn | dyn\_partition

Indicates in which static or dynamic partition the specified address is to be displayed. You can specify any of the static partitions BG, F1 through FB or a partition within a dynamic class, for example, P1.

#### jobname

Indicates the job name of the job to be displayed. *jobname* can be up to 8 characters and must be unique.

#### address

Specifies the address at which the storage display is to start. **address** can be a 1- 8-digit hexadecimal address. The highest address that can be specified is limited by the size of the shared areas plus the size of the private area (SYS PASIZE value).

If space-id S has been specified and the specified address is not within the shared area (supervisor, SVA or shared partitions), the command is ignored and a corresponding information message is issued. If the specified space-id is one of 0 through 9, A or B, any address between 0 and end-of-storage is accepted.

If the specified address is within a dynamic partition, the corresponding dynamic space GETVIS area can be displayed, too.

If the specified address is within an invalid address area, the command is ignored and a corresponding information message is issued.

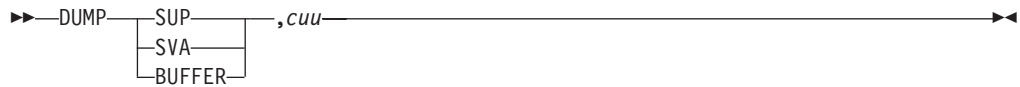
If the 16 bytes to be displayed cross the boundary from a valid to an invalid address area, only the bytes in the valid address area are displayed, and a corresponding information message is issued.

## DUMP

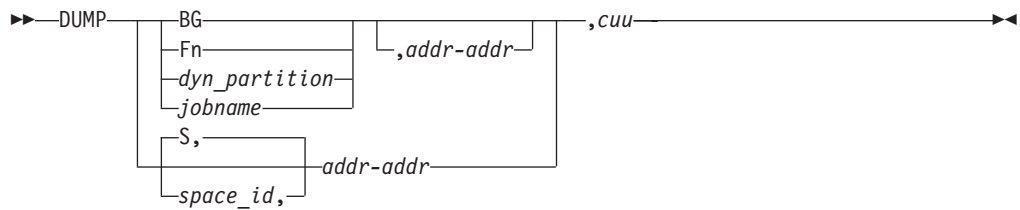
### DUMP (Dump Storage Areas)

The DUMP command allows the operator to dump specified areas of virtual address space or data space storage on a printer or tape device.

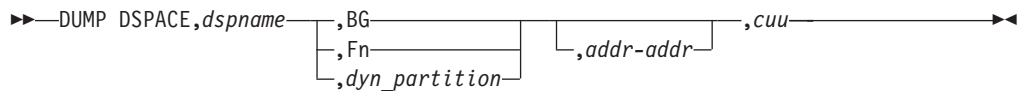
#### AR Format



#### AR Format



#### AR Format



#### SUP

Dumps the supervisor area and the control registers.

#### SVA

Produces a dump of either the whole Shared Virtual Area or selected parts of it, or a single phase within the SVA.

The system responds with message

1I59D ENTER PHASE NAME, SVA24, GETVIS24, SVA31, GETVIS31 OR ALL

The operator may enter one of the following:

- The name of an SVA phase
- SVA24 for the Shared Virtual Area (24)
- SVA31 for the Shared Virtual Area (31)
- GETVIS24 for the system GETVIS area (24)
- GETVIS31 for the system GETVIS area (31)
- ALL for the whole SVA and system GETVIS area

#### BUFFER

Writes the contents of the SDAID buffer to tape. This operand is accepted only if the dump is directed to a tape device.

#### cuu

Specifies the device on which the output is to be written. It can be a printer or tape device, unless BUFFER was specified in the first operand. In this case, only a tape unit address is accepted. Tape output is written without repositioning the tape to allow for several dumps per tape. The output is written after the preceding DUMP command output to allow for several

dumps per file. For information on dump handling, refer to “Part 1. Dumps of Virtual Storage” in the *z/VSE Diagnosis Tools*.

If *cuu* designates a printer, the printer should not, at the time of the dump, be used by the partition to which it is assigned, since this could result in interspersed partition and dump output.

**Note:** When *cuu* is assigned to a 3800 Printing Subsystem, you must ensure that the 3800 settings are appropriate for the expected output.

### **BG | Fn | dyn\_partition**

Specifies the partition identifier of the (static or dynamic) partition to be dumped. You can specify any of the static partitions BG, F1 through FB or a partition within a dynamic class, for example, P1. The DUMP command produces a dump of the PSW, the general, floating-point, and access registers from the partition save area and the specified partition area, **excluding** the dynamic space GETVIS area. To get a dump of the dynamic space GETVIS area, do the following:

1. Enter MAP *xx* where *xx* is the partition ID (for example, P1)
2. Determine the start and end addresses of the dynamic space GETVIS area from the MAP output
3. Dump the area with the following command:

```
DUMP XX,addr-addr,cuu
```

where:

**XX** is the partition ID (P1)

**addr-addr** is the starting and ending address of the dynamic space GETVIS area

**cuu** is the output device, either a tape drive or a printer.

### **jobname**

Indicates the job name of the job to be dumped. *jobname* can be up to 8 characters and must be unique.

### **space\_id**

Specifies the address space of the storage area to be dumped. Valid specifications are:

R (real) or S (shared)

0 through 9, A, B

If the space identifier is omitted, S is assumed as default; that is, only those portions of storage are dumped which are part of shared spaces.

### **addr-addr**

Dumps the virtual storage between the specified addresses either in the partition indicated by **part** (default: whole partition) or in the address space indicated by **space-id**. If any active real or virtual partition, or a part of such a partition, lies between the specified addresses, its PSW and associated registers are dumped.

The last format dumps selected areas of **data space** storage.

### **DSPACE**

Produces a dump of the data spaces.

## DUMP

### **dspname**

Specifies the data space name which may be one to eight characters long. The operator can retrieve the data space name via the QUERY DSPACE command. See Note.

### **BG | Fn | dyn\_partition**

Specifies the partition which owns the data space to be dumped. See Note.

### **addr-addr**

Defines the start and end address of the storage area within the data space to be dumped. If the operand is omitted, the whole data space is dumped.

### **cuu**

Specifies the device on which the output is to be written.

**Note:** Data space names are unique only within a partition; they need not be unique within the system. Different partitions may own data spaces with the same name.



## DVCDN (Device Down)

The DVCDN command informs the system that a device is no longer available for system operation. It is used when a device is to be serviced or becomes inoperative.

This command may be given in any partition, and the specified device is made unavailable for all partitions.

### JCC Format

▶▶—DVCDN *cuu*—————▶▶

#### **cuu**

Indicates the device number of the device to be made unavailable.

**Note:** The system does not accept the *cuu* of a device on which SYSRES, SYSREC, SYSCAT, the internally assigned system logical unit SYSDMP or the page data set resides.

If a permanent or temporary assignment exists for the device specified in the command, any logical units assigned to it are unassigned.

Access to the device is only possible via physical IOCS (PIOCS) operations.

If a sublibrary on the specified device is part of a sublibrary chain (specified in a LIBDEF statement), the DVCDN command is not accepted.

The DVCDN command does not close files associated with logical units, and after the DVCDN command has been issued, files on a disk or diskette unit cannot be closed or reassigned to another disk or diskette unit. Therefore, if the unit is a disk or diskette unit, first attempt to close any files associated with logical units currently assigned to the device, using the CLOSE command.

If an alternate assignment exists for the device, it is removed when the DVCDN command is issued. A DVCUP command must be issued before the device can be used again. See also the OFFLINE command on page OFFLINE (Simulate DEVICE OR CHPID NOT READY) on page 160.

### DVCUP (Device Up)

The DVCUP command informs the system that a device which was inoperative is now available again for system operation. As all assignments for this device were removed by the preceding DVCDN command, the device must be reassigned by an ASSGN statement or command.

Note that the DVCUP command is ignored for CMS disks. They stay in device down status. No error message is issued.

The command is not allowed

- For a virtual disk that has not been defined with the VDISK command (but only added with the ADD command).
- For the secondary device of a duplex pair of disks.
- For a device with type code ESCD (ESCON Director).

### JCC Format

►►—DVCUP *cuu*——————►►

#### **cuu**

**c** is the channel number and **uu** the unit number, in hexadecimal, of the device to be made available.

**END or ENTER (End of Input)**

The END or ENTER command must be issued whenever the operator has finished typing an input line. It causes the communication routine to return control to the mainline job. END applies to CPU models with a printer keyboard console. ENTER applies to CPU models with a display console.

**JCC, AR Format**

Press the END or ENTER key.

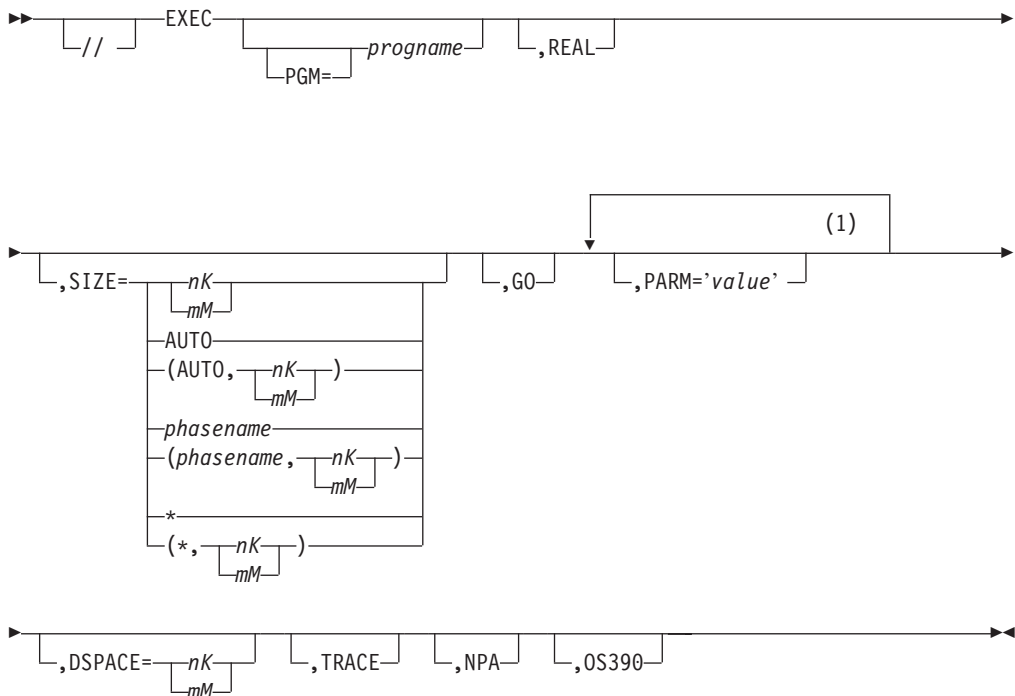
## EXEC (Execute Program or Procedure)

The EXEC command or statement indicates either:

- The end of control information for a job step and the beginning of execution of a program.
- That a cataloged procedure or REXX procedure is to be retrieved from a sublibrary by job control.

**Note:** If the access control function is active, and a program is to be executed while a protected sublibrary is part of the LIBDEF PHASE,SEARCH chain in the partition, this sublibrary must be opened so that authorization checking can be done. This means that the labels for this library must be available in the label information area when the EXEC statement is issued.

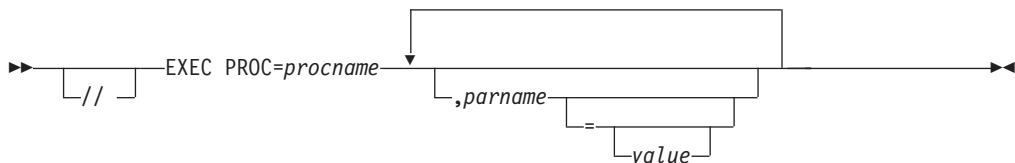
### Format 1 (JCS, JCC)



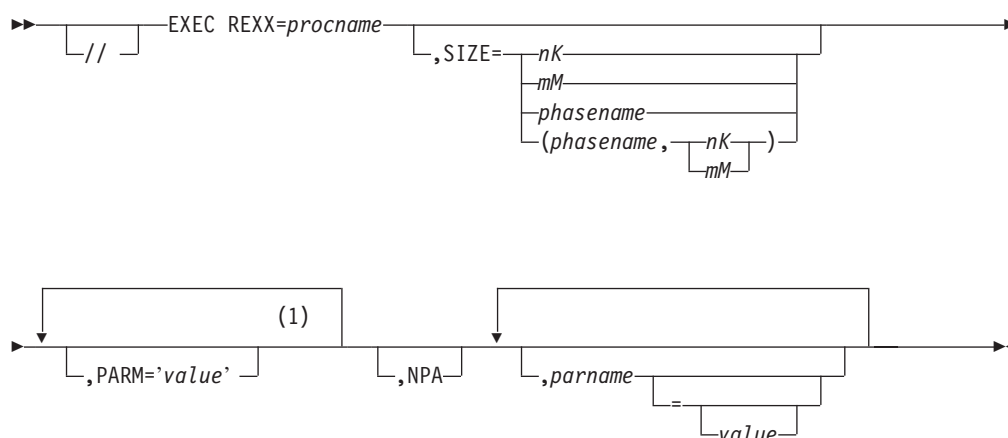
#### Notes:

- 1 Up to and inclusive the PARM operand the operands must be entered in the specified order. PARM= can be specified up to three times.

### Format 2 (JCS, JCC)



### Format 3 (JCS, JCC)



#### Notes:

- 1 PARM= can be specified up to three times.

The statement or command with a **program** name can be issued from SYSLOG or from SYSRDR. Control returns to the unit from which the statement or command was issued.

With a **procedure** name, the statement can be issued from SYSLOG or from SYSRDR; control always returns to SYSRDR. The command with a procedure name can be issued from SYSLOG only.

Continuation lines are accepted for the EXEC statement.

### Format 1

#### PGM=programe

Represents the name of the program to be executed. The program name corresponds to the first or only phase of the program in the library. If the program to be executed has just been processed by the linkage editor, the program name is omitted and the PGM keyword cannot be used.

#### REAL

Indicates that the program will be executed in real mode. If REAL is not specified, the program is always executed in virtual mode.

The operand is not allowed in a dynamic partition, in which case an error message is issued.

#### SIZE=

The SIZE operand can be specified in combination with REAL or without REAL.

1. If specified **with** REAL, it gives the size of that part of the partition's processor storage that will be needed by the program. The remaining part of the allocated processor storage can be used as additional storage (GETVIS area) for other modules or data required by the program in that partition. The program obtains this additional storage by issuing GETVIS macros with the required amount of storage as an operand; it releases the storage by issuing FREEVIS macros.

If the SIZE operand is omitted and REAL is specified, the entire processor storage of the partition is reserved for the program.

## EXEC

- If used **without** REAL, it specifies the size of that part of the virtual partition that will be directly available to the program. The remainder of the partition may be used as additional storage (GETVIS area) for other modules or data required by the program in that partition. The system always allocates a minimum partition GETVIS area of 48KB.

Certain programs have partition GETVIS requirements beyond 48KB, such as VSE/VSAM programs, ISAM programs using the ISAM Interface Program (IIP), programs using RPS support, or programs using sequentially organized disk files. Through the SIZE operand, you can temporarily change the size of the partition GETVIS area. The new GETVIS area size is the total partition size minus the value specified in the SIZE operand. The SIZE specification is accepted only if it yields a GETVIS area larger than 48KB.

If the SIZE (and the REAL) operand is omitted, either the whole virtual partition minus the minimum GETVIS area, or the default GETVIS area as specified by a preceding SIZE command, is reserved for the job initiated with EXEC.

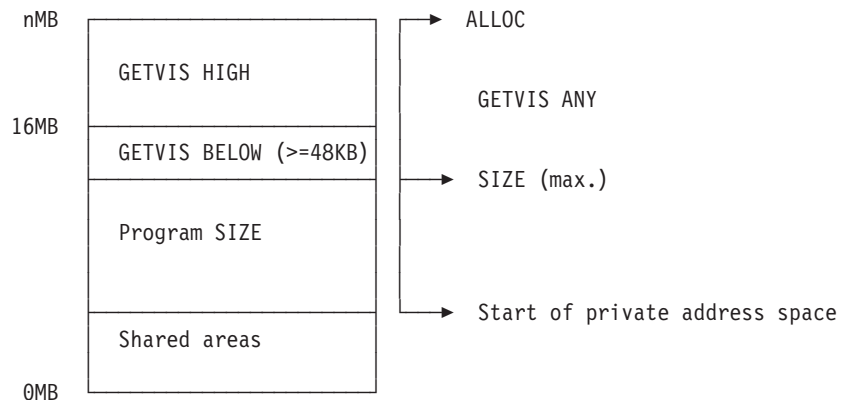
The SIZE operand can be specified in the following formats:

```
SIZE={nK|mM}
SIZE=AUTO
SIZE=(AUTO, {nK|mM})
SIZE=phasename
SIZE=(phasename, {nK|mM})
SIZE=*
SIZE=(*, {nK|mM})
```

**Note:** If this operand is not AUTO, \*, nK or nM, the system assumes that it is a phase name.

**n** or **m** must be greater than zero and **n** must be a multiple of 4 (if not, the system rounds the value up to the nearest 4K boundary).

**nK** or **mM** must not exceed the size of the partition (as defined by ALLOC) minus the minimum partition GETVIS area of 48KB. Since the SIZE definition (in any format) must not cross the 16MB line, the system ensures that the start of the partition GETVIS area is not moved beyond the (16MB minus 48KB)-line:



$SIZE(max) = 16MB - 48KB(min. GETVIS BELOW) - \text{size of shared areas}$

**Note:** If you specify a SIZE which is less than the storage which the program in fact requires, the GETVIS area may be overlaid, with unpredictable results.

**AUTO** indicates that the program size, as calculated by the system from information in the sublibrary directory, is to be taken as the value for **SIZE**. Use caution in specifying **SIZE=AUTO** in the following case: When phases belonging to the same program (multi-phase) or same application (for example, payroll) use generic phase names (identical first four characters), the size of the phase with the highest ending address found with that generic name will be used.

**Note:** Do not specify **SIZE=AUTO** for programs that dynamically allocate storage during execution (such as linkage editor, librarian program, and compilers).

**AUTO,{nK|mM}** indicates that job control must take the program size plus nK or mM bytes as the value for **SIZE**.

**phasename** indicates that the length of the specified phase, increased by its relative load address in the partition, is to be taken as the value for **SIZE**, regardless of other phases with the same first four characters in their names.

**phasename,{nK|mM}** indicates that the length of the specified phase, increased by its relative load address in the partition, plus nK or mM bytes, is to be taken as the value for **SIZE**. If this value is not a multiple of two, it is rounded up.

**\*** indicates that the length of phase proname (specified in **PGM=**), increased by its relative load address in the partition, is to be taken as the value for **SIZE**, regardless of other phases with the same first four characters in their names.

**\*,{nK|mM}** indicates that the length of phase proname (specified in **PGM=**), increased by its relative load address in the partition, plus nK or mM bytes, is to be taken as the value for **SIZE**. If this value is not a multiple of two, it is rounded up.

**Note:** Do not specify **SIZE=AUTO** or **SIZE=\*** for programs that dynamically allocate storage during execution (such as linkage editor, librarian program, and compilers).

## GO

Specifies, for a language translator step, that the program is to be link-edited and executed automatically after it has been compiled. Only the source program data and any additional input data for the execution step are required after the language translator step. If a serious error is encountered in either the language translator step or the link-edit step, the input stream is flushed to the end-of-job (/&) statement.

**Note:** This type of execution can be used only for single-phase programs.

## PARM='value'

Specifies information which is to be passed to the program at execution. **value** can be up to 100 characters in length, enclosed in quotes. (The enclosing quotes are not passed to the program.) An quote within **value** must be coded as two single quotes. If you need to pass a parameter value that is longer than 100 characters, you can code **PARM='value'** up to three times on one EXEC statement. The syntax rules above apply to each PARM operand separately.

The information given by value is stored into the system GETVIS area. If **PARM='value'** was specified twice or three times, the values are concatenated according to their sequence.

For information on how to access the PARM value from an assembler program, see "Communicating with Application Programs via Job Control" in the *z/VSE Guide to System Functions*.

#### **DSPACE=nK | mM**

Specifies, e.g. for VTAM<sup>®</sup> applications, the maximum size of a data space, where **n** or **m** must be greater than zero and **n** must be a multiple of 4 (if not, the system rounds the value up to the higher 4K boundary). As a data space cannot be larger than 2 Gigabytes, an error message is issued if **n** exceeds 2,097,148 or **m** exceeds 2,047. The system does not validate or use the stored value; that is, the application is responsible that the requested data space size is available. Also, the values **n** and **m** are not validated against the DSPACE parameter values of the SYSDEF command.

#### **TRACE**

Activates the interactive trace program for the user program named **progname**. The invoked trace function is active for the duration of one VSE job step. TRACE defines an instruction trace and an ABEND trace. These trace definitions allow the console operator to get interactive control over the program to be traced. The instruction trace passes control to the console operator at the beginning of a user program, the ABEND trace allows debugging when a program terminates abnormally.

#### **NPA**

In particular problem situations it may be desirable to interfere and enforce non-parallel processing for a program. For this purpose the NPA (Non-Parallel Application) operand is available. It should be used for problem solving only in the following cases:

- A program runs correctly on a uniprocessor but not on a multiprocessor.
- Two programs run and communicate correctly on a uniprocessor but not on a multiprocessor. This may be a synchronization problem.

The NPA operand is meaningful only if the following conditions exist:

- The program or application is not a key 0 program (key 0 programs are usually system programs).
- At least one additional CPU has been started.

If these conditions do not exist, the NPA operand is ignored.

#### **OS390**

This operand requests OS/390<sup>®</sup> emulation mode, which is required, for example, for the CICS Transaction Server. This mode allows the execution of emulated OS/390 services. This operand can only be used when one single partition is allocated in an address space (single partition allocation). It will be rejected in case of multiple-partition (ALLOC space\_id), real (ALLOC R) or shared (ALLOC S) allocations.

### **Format 2**

#### **PROC=procname**

Represents the name of the procedure to be retrieved from a sublibrary.

If the procedure name begins with \$\$, the system substitutes, for static partitions, a partition-related character for the second \$. The character that is substituted is related to the static partition in which the procedure is invoked, that is,



0 for the BG partition  
 B for the FB partition  
 A for the FA partition  
 9 for the F9 partition  
 . . .  
 1 for the F1 partition.

For a dynamic partition, the **first** \$ sign is replaced by the dynamic **class** of the partition, that is, the first character of the partition identifier.

The procedure corresponding to this name is then retrieved for execution.

#### **parname=value**

There are three methods of addressing symbolic parameters in the EXEC PROC or EXEC REXX statement or command:

1. parname1=value1
2. parname2
3. parname3=&parname3

#### **parname1**

Specifies the name of a symbolic parameter which is to be substituted in the specified procedure. It must consist of 1 to 7 alphanumeric (including national) characters, and the first character must be alphabetic. The & at the beginning of the symbolic parameter as coded in the called procedure must not be coded in the EXEC statement. For example, if you want to substitute the symbolic parameter &PARM1 with the value PAYROLL, you must code PARM1=PAYROLL.

#### **value1**

Specifies the actual value which is to be inserted in the specified procedure in place of the specified symbolic parameter. It must be a string of up to 50 characters. If the string is alphanumeric, no enclosing quotes are necessary. If it contains national or special characters, it must be enclosed in quotes, which will not be passed to the procedure. No quotes are allowed in the string itself.

If value1 is a null string (PARM1=" or PARM1=), the specified parameter is ignored in the called procedure.

#### **parname2**

Is the name of a symbolic parameter which is to be passed to a lower-level procedure and back. The value assigned to the parameter on the higher JC level at the time of the call will be valid for the lower-level (called) procedure, and if it is altered in the lower-level procedure by SETPARM, the new value will also be valid for the higher JC level on return.

#### **parname3**

Is the name of a symbolic parameter which is to be passed to a lower-level procedure. The value assigned to the parameter on the higher (calling) JC level at the time of the call is valid for the lower level (called) procedure, but can be altered there with no effect on the corresponding parameter on the higher level. The symbolic parameter name after the equals sign must be coded *with* the ampersand (&).

### **Format 3**

#### **REXX=procname,...**

Indicates the name of a procedure in a sublibrary that is to be executed by REXX.

## EXEC

If the procedure begins with \$\$, the system substitutes the second \$ in the same way as described above under PROC=procname. (All rules that apply to calling JCL procedures - such as data mode, nesting - also apply to calling REXX procedures.)

The procedure corresponding to the specified name is then accessed, but **not** executed by Job Control. The PARM value (if any) is passed to REXX and REXX retrieves and executes the procedure.

The **SIZE** operand specifies the size of the program area used by REXX to load the programs that do not run in the GETVIS area. In addition, 80K bytes are added for Job Control itself.

After REXX has finished, it may return job control statements that were queued on the REXX stack back to Job Control. These statements are then processed sequentially under the procedure name **procname** specified in the EXEC statement.

A stacked procedure produced by a REXX procedure that was cataloged with the DATA=NO option (default) reads SYSIPT data (if needed) from the device assigned to SYSIPT.

A stacked procedure produced by a REXX procedure that was cataloged with the DATA=YES option must contain all SYSIPT data needed. For example, if the procedure contains an EXEC statement for a program reading data from SYSIPT, then all SYSIPT data must immediately follow that EXEC statement (as in normal JCL procedures with DATA=YES).

### **parname=value**

For a description of *parname=value*, see page 103. The symbolic parameters can be used in JCL statements issued by the ADDRESS JCL command environment and in the stack passed to Job Control when REXX terminates.

You must **not** use SIZE, PARM, or NPA as symbolic parameters names together with EXEC REXX.

## EXPLAIN (Online Explanation Support)

The EXPLAIN command allows to activate and deactivate Online Message Explanation support, and to query its current status.

### AR Format



- ON** Indicates that EXPLAIN support is to be activated, and causes the Online Message Explanation file to be opened, if not already open.
- OFF** Indicates that EXPLAIN support is to be deactivated, and causes the Online Message Explanation file to be closed, if currently open.

When entered without parameter, the current status, ON or OFF, is displayed.

The initial status after IPL is OFF. EXPLAIN ON may be included in the BG ASI procedure, to activate the support as part of the IPL process, after label definition for the message explanation file (VSE.MESSAGES.ONLINE) has been done. EXPLAIN ON is already set in the standard z/VSE procedure \$0JCL.

## EXTENT (Disk or Diskette Extent Information)

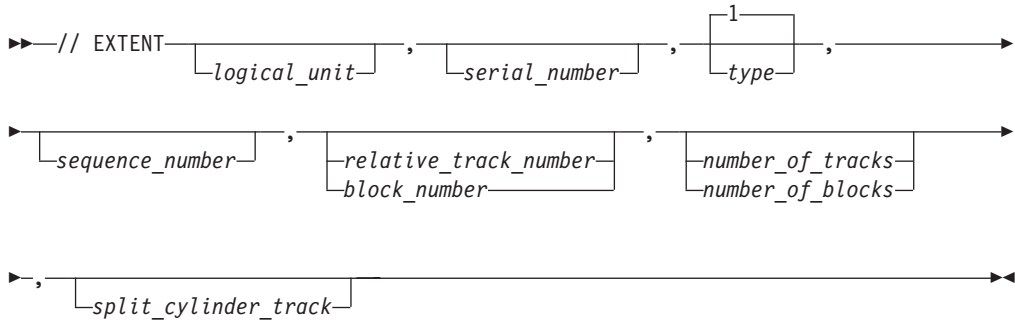
The EXTENT statement defines each area, or extent, of a disk or diskette file. One or more EXTENT statements must directly follow each DLBL statement, except for VSAM files and for single-volume input files for sequential disk on a disk or diskette, provided the DEVADDR parameter has been specified in the DTF table.

**Note:** The EXTENT statements should be checked carefully because an invalid field causes the default options or the values entered by the previous EXTENT statement to be overwritten by the valid entries of the flagged statement.

System files on disk (SYSIPT, SYSRDR, SYSLST, SYSPCH) and SYSLNK (always on disk) must have only **one** extent.

Multiple EXTENT statements are valid for system files on diskette. Valid parameters are logical unit, serial number, and type. The other parameters will be ignored.

### JCS Format



No comma need be coded for an EXTENT statement without any operands.

#### logical\_unit

A six-character field indicating the logical unit (SYSxxx) of the volume for which this extent is effective. If this operand is omitted, the logical unit of the preceding EXTENT, if any, is used. If this operand is omitted on the first or only EXTENT statement, the logical unit specified in the DTF is assumed. A logical unit included in the extent information for SAM, DAM, ISAM, or diskette files, however, overrides the DTF DEVADDR=SYSnnn specification.

This operand is not required if a system file with IJSYSxx as filename is specified. The following IJSYSxx filenames in a DLBL statement cause their corresponding default logical units to be specified in the EXTENT statement:

File Name	Default Logical Unit
IJSYSIN	SYSIN (SYSRDR/SYSIPT)
IJSYSPH	SYSPCH
IJSYSLS	SYSLST
IJSYSLN	SYSLNK
IJSYSRS	SYSRES
IJSYSxx	SYS0xx

The operand is also optional for a user file defined with a DTF DEVADDR=SYSnnn. If SYSRDR or SYSIPT is assigned, this operand must be included.

In multivolume SAM, DAM, ISAM, and diskette files, each different logical unit must be assigned to a separate physical device. In multi-extent SAM, DAM, ISAM, and diskette files, all extents on one physical unit must have the same logical unit number.

For SAM and DAM files, both logical unit and sequence numbers must be in consecutive ascending order.

User programs may use, in addition to programmer logical units, the following system logical units:

SYSIPT and SYSRDR for input

SYSLST and SYSPCH for output

#### **serial\_number**

From one to six characters indicating the volume serial number of the volume for which this extent is effective. If fewer than six characters are used, the field is padded on the left with zeros, unless you enclose it in quotes, in which case it is padded on the right with blank characters.

If this operand is omitted, the volume serial number of the preceding EXTENT is used. Therefore, when a multivolume file is being processed, the volume serial number of the first volume is assumed for the entire file, unless you specify this field for the first extent of each following volume. If no serial number was provided in the EXTENT statement, the serial number is not checked and it is your responsibility if files are destroyed because the wrong volume was mounted. The serial number must be specified for VSE/VSAM file extents.

For the diskette, this operand specifies that the associated file will be found on this volume. If the parameter is omitted, the OPEN routines assume that the volume that was mounted is the correct one. Label checking will be done for input files and space will be allocated for an output file.

One EXTENT statement must be submitted for each volume of an input file, and sufficient EXTENT statements must be submitted for output files to ensure that enough volumes are present to contain the file.

#### **type**

One character indicating the type of the extent, as follows:

- 1 data area (no split cylinder)
- 2 independent overflow area (for indexed sequential files)
- 4 index area (for indexed sequential files)
- 8 data area (split cylinder, for SAM files only, but not on FBA devices)

If this operand is omitted, type 1 is assumed. 1 is the only valid type specification for diskette files.

For indexed sequential files, enter the extent information in the following order:

1. Master index (type 4) and sequence number 0.

## EXTENT

2. Cylinder index (type 4) and sequence number 1.
3. Prime data area (type 1) and sequence number 2, 3, ..., n.
4. Independent overflow area (type 2) and sequence number (n+1).

where n is the sequence number of the last prime data area extent.

Note also that the master and the cylinder index must be in adjacent areas on the same logical unit.

### **sequence\_number**

One to three characters containing a decimal number 0 to 255 indicating the sequence number of this extent within a multi-extent file. Extent sequence number 0 is used for the master index of an indexed sequential file. If the master index is not used, the first extent of an indexed sequential file has the sequence number 1. The extent sequence number for all other types of files begins with 0. If this operand is omitted for the first extent of ISAM files, the extent is not accepted. For SAM and VSE/VSAM files, this operand is not required. This parameter is ignored for diskette files. For SAM and DAM files, both logical unit and sequence numbers must be in consecutive ascending order.

### **relative\_track\_number | block\_number**

For **CKD** devices, this operand is one to six characters indicating the sequential number of the track, relative to zero, where the data extent is to begin. If this field is omitted on an ISAM file, the extent is not accepted. This field is not required for SAM input files (the extents from the file labels are used). This field must be specified for DAM input files.

When using split cylinder files, this parameter designates the beginning of the split as well as the first track of the file.

To convert an actual address (in cylinders and tracks) to a relative track address, and vice versa, use the following formulae:

#### **Actual to Relative**

$$T/C \times \text{cylinder number} + \text{track number} = RT$$

#### **Relative to Actual**

$$RT : T/C = \text{cylinder number,} \\ \text{remainder is track number}$$

where RT is the relative track, and T/C is the number of tracks per cylinder for the device type in question, as shown in Table 7.

Table 7. Number of Tracks per Cylinder for Disk Devices

IBM Device Type	Tracks per Cylinder
3380	15
3390	15

**Example:** Track 5, cylinder 150 on a 3380 = relative track 2255.

For **FBA** devices, this operand is a number from 2 to 2,147,483,645 which specifies the physical block at which the extent is to start.

For **VSE/VSAM**, this operand must be specified when a data space or a file with the **UNIQUE** option is being created. This operand is not required, and it is ignored if it is specified, when a VSE/VSAM file is created within an existing data space. In this case, the space for the file is sub-allocated by

VSE/VSAM from direct-access extents it already owns. This operand is also not required for VSE/VSAM input files because the extents are obtained from the VSE/VSAM catalog.

**number\_of\_tracks | number\_of\_blocks**

For **CKD** devices, this operand is one to six characters indicating the number of tracks to be allocated to the file. For SD input, this field may be omitted, provided the 'relative track' field is also omitted. For an indexed sequential file, the number of tracks for prime data must be a multiple of the number of tracks per cylinder of the disk device used. For details, see Table 7 on page 108.

The number of tracks for a split cylinder file must be the product of the number of cylinders for the file and the specified number of tracks per cylinder for that file.

For **FBA** devices, this operand is a number from 1 to 2,147,483,645 which specifies the number of physical blocks in the extent.

This operand and **relative\_track\_number** or **block\_number** must either both be present or both be omitted. If the operands are present in an initial EXTENT statement, they must also be specified in all succeeding EXTENT statements. If they are omitted, they are ignored in all succeeding EXTENT statements.

**split\_cylinder\_track**

A one or two-digit decimal number, indicating the upper track number for the split cylinder in SAM files (for CKD devices only). The minimum specification is 0, the maximum is device-dependent, and is 1 less than the number of tracks per cylinder (see Table 7 on page 108) for the device in question.

## FREE

### FREE (Reset RESERV Command)

The FREE command is used to reset the RESERVED status (as caused by the RESERV command) of the specified device. The command may be issued for all disk devices on the system.

#### AR Format

▶▶—FREE *cuu*—————▶▶

#### **cuu**

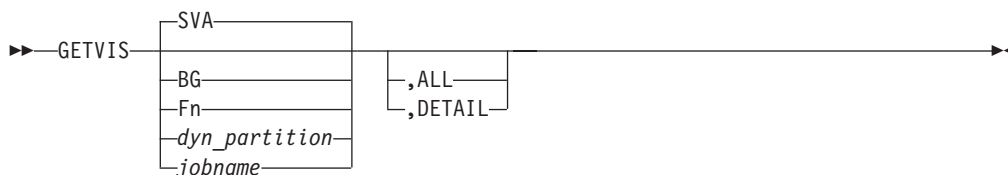
Indicates the device number of the device to be freed.



## GETVIS (Display GETVIS Information)

The GETVIS command displays information about the **current** size, allocation, and usage of the GETVIS area of a static or dynamic partition or of the system GETVIS area.

### AR Format



#### SVA

Specifies that you want the system to display information about the system GETVIS space in the shared virtual area.

#### BG | Fn | dyn\_partition | jobname

Specifies the partition for which you want the system to display GETVIS space related information. You can specify any of your system's active static partitions BG through FB or any dynamic partition. The "Display Storage Layout" panel of z/VSE shows the layout of the dynamic space GETVIS area for dynamic partitions. You can also specify the name of the job.

#### ALL

Indicates that a summary report of the subpools currently active is to be displayed. This report will provide information about the storage in K-Bytes (1024 Bytes) that is currently held available or is already in use by a certain subpool. The report distinguishes between GETVIS-24 versus GETVIS-ANY storage.

#### DETAIL

Indicates that a detailed report of any of the pages of any of the subpools currently active is to be displayed. This report will provide information about any of the adjacent storage locations that are currently held available or are already in use by a certain subpool. The report distinguishes between GETVIS-24 versus GETVIS-ANY storage.

Following is an example of a display of a static partition-GETVIS information:

```
AR 015 1140I  READY
=> GETVIS F2
AR 015 GETVIS USAGE   F2-24   F2-ANY           F2-24   F2-ANY
AR 015   AREA SIZE:   4.096K   7.168K
AR 015   ALLOCATED:    100K    228K  MAX. EVER USED:  104K   148K
AR 015   FREE AREA:   3.996K   6.940K  LARGEST FREE:    3.992K  6.936K
AR 015 1140I  READY
```

In summary, this display tells the following:

- The partition has an actual GETVIS space of 4.096 KB below 16MB (F2-24) and a total GETVIS space of 7.168KB (F2-ANY). (F2-ANY) - (F2-24) gives the GETVIS space for the 31-Bit area.
- 100 KB (of 4.096) are currently allocated below 16MB; 128 KB are used above 16MB, that means a total of 228KB is used (of 7.168KB).

It may be that a MAX.EVER USED F2-24 shows a total of 4.096KB and F2-ANY a total of 7.168KB. There may be no unique reason for this situation. It could be that both the space above 16MB and below 16MB really is used up.

## GETVIS

Definitely the area above 16MB is exhausted. However, if a LOC=ANY request was redirected to the area below 16MB, the MAX.EVER USED value is set to a maximum although the area below 16MB is not completely used up. In this case increase the partition size (area above 16MB) and check the GETVIS usage again until the situation disappears.

- 104KB (respectively 148KB) is the highest number of bytes that has been used as GETVIS space at any point in time since you started the partition (high watermark).
- The largest contiguous free area has a size of 3.992KB (6.936KB).

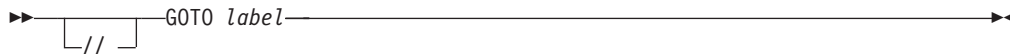
In addition to the system GETVIS and partition GETVIS area a dynamic partition also includes a dynamic-space GETVIS area as shown in the example below:

```
AR 015 1I40I  READY
=> GETVIS M1
AR 015 GETVIS USAGE      M1-24      M1-ANY                M1-24      M1-ANY
AR 015  AREA SIZE:      3.984K      3.984K
AR 015  ALLOCATED:        32K         32K  MAX. EVER USED:    32K      32K
AR 015  FREE AREA:      3.952K      3.952K  LARGEST FREE:    3.952K    3.952K
AR 015 DYNAMIC -SPACE GETVIS USAGE
AR 015  AREA SIZE:        256K
AR 015  ALLOCATED:         24K          MAX. EVER USED:    36K
AR 015  FREE AREA:       232K      3.952K  LARGEST FREE:    232K
AR 015 1I40I  READY
```

## GOTO (Skip to Label)

The GOTO statement causes all statements in the following job stream to be skipped, up to the specified label statement. It is accepted only within a job.

### JCC, JCS Format



### label

Specifies the operand of the /. statement at which execution of the current job is to continue. Code \$EOJ to skip all statements up to end-of-job.

The job stream cannot be searched backwards, and the target label statement must be on the same level as the GOTO statement, that is, both outside a procedure or both in the same procedure.

JC does not check for duplicate labels. If two or more label statements are coded with the same operand, execution will continue after the first one to be found.

All JCL statements between the GOTO and the target label statement are ignored, except for the following:

- Statements entered from SYSLOG.
- /+ (End-of-Procedure) - if this is encountered before the specified label statement is found, the rest of the job is skipped.
- // JOB and /& statements - if these are encountered, the job and its called procedure(s) are terminated.

**Note:** If you enter the GOTO statement from SYSLOG, the system gives you the opportunity to enter further statements (except GOTO and EXEC PROC) from SYSLOG. When you enter a blank line, JC switches back to SYSRDR and searches for the specified label.

## HCLOG (Control Message Logging)

The HCLOG command allows to control the scope of messages logged on the hardcopy file.

### AR Format



#### ALL

Indicates that all console traffic is to be logged, except for DOM requests and for Redisplay commands and responses (see the DOM macro).

#### MASTER

Indicates that logging is limited to

- Messages that are routed to master consoles, or CMS users with MASTER authority,
- All input from such consoles or CMS users,
- Command input from all consoles,

with the same exceptions as for ALL.

If the command is entered without operand, the current setting of the logging option is displayed.

## HOLD (Hold Assignments and LIBDEFs)

The HOLD command is used to hold assignments or sublibrary definitions (LIBDEF) before you issue a command to unbatch a foreground partition. The partitions may be specified in any sequence; at least one partition must be given.

The command is not allowed in a dynamic partition.

### JCC Format



**n** indicates the desired partition.

## ID (User-ID and Password)

The ID statement or command is used to specify the user identification and the user's password. This information is checked against the contents of the user profile and, depending on the result of this check, the job is allowed to run or it is canceled.

An ID statement or command is required if a job uses resources protected by the access authorization facility of VSE. The // ID statement must be specified after the JOB statement; it is valid until end-of-job or until a subsequent // ID statement is specified within the same job.

### JCC, JCS Format



#### USER=user-id

Specifies the user identifier, which can be four to eight alphanumeric characters.

#### PWD=password

Specifies the password of the user, which can be three to eight alphanumeric characters.

Neither the user-id nor the password will be displayed on SYSLOG or SYSLST. If the ID statement/command causes an error message, it is logged in the following format to avoid disclosure of the password:

```
ID (PARAMETERS SUPPRESSED)
```

## IF (Check Local Condition)

The IF statement is a local conditional function. When it occurs in the job stream, the specified condition is checked. If it is true, the following statement is executed; if not, the following statement is skipped.

The IF statement is accepted only within a job.

Continuation lines are accepted for the IF statement.

### JCC, JCS Format



#### condition

Specifies a condition to be checked. It may be expressed in one of the following forms:

```

$RC comparator n
$MRC comparator n
pname comparator value
  
```

where:

#### \$RC

Specifies the return code of the preceding job step.

#### \$MRC

Specifies the maximum return code of all preceding steps within the current job.

#### pname

Specifies the name of a parameter to be compared.

#### comparator

Specifies the comparison to be done. This can be one of the following six possibilities:

Comparison:	Specified as:
Equal to	= or EQ
Not equal to	≠ or NE
Greater than	> or GT
Less than	< or LT
Greater or equal	>= or GE
Less or equal	<= or LE

If both comparands are numbers, the system does an arithmetic comparison. If one or both of the comparands contains any non-numeric character, the system does a logical comparison. This is carried out in the length of the longer comparand, and the shorter comparand is padded on the right with blank characters. If one of the comparands is a null string, only the comparators =, ≠, EQ and NE are accepted.

**n** Specifies a decimal integer from 0 to 4095.

#### value

Specifies a character string of 0 to 50 characters. If the string contains special characters, it must be enclosed in quotes. No quotes are allowed within the string. You can, of course, specify a symbolic parameter for this operand, for example,

## IF

```
IF PARM1>&PARM2 THEN
```

Or you can use a null string, For example:

```
IF PARM1='' THEN
```

### **operator**

Specifies a logical operator which connects two conditions. The valid specifications are: OR, |, AND, &; The statement following the IF statement is executed when:

- The conditions are connected by OR or |, and one or both of them is true,
- The conditions are connected by AND or &, and both of them are true.

The logical operators OR, |, AND, & must be preceded and followed by a blank character.

You may enter an IF command from the console. In this case, the “following statement” is the next command you enter from the console, or the next statement from SYSRDR, if you enter a null line (just press END or ENTER) at the console.

**Note:** If the statement following the IF is a JOB, /& or /+ statement, it is not skipped, even when the condition in the IF statement is false.

For an example of the use of the IF statement, see Figure 18 on page 261.



## IGNORE (Ignore Abnormal Condition)

Whenever an abnormal condition arises, the operator will be notified by an appropriate message on SYSLOG. Depending on the situation, he may have to ignore the condition by entering an IGNORE command. This is indicated under "Operator Action" in *z/VSE Messages and Codes* for each applicable message.

### JCC, AR Format

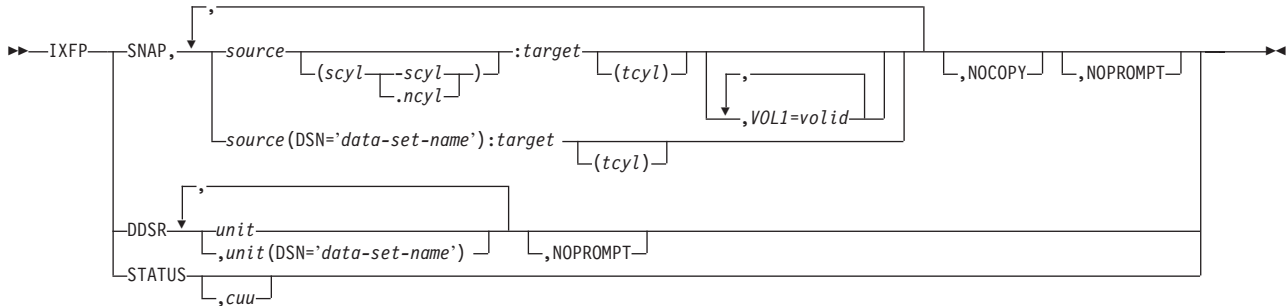
▶—IGNORE—▶

The IGNORE command has no operand.

## IXFP SNAP

IXFP SNAP copies data from a source device to a target device. Using this command, the FlashCopy functionality is invoked.

### JCC, AR Format



**SNAP** The IXFP SNAP function copies data from a source device to a target device.

*source* The device ID (cuu) or the VOL1 label of the SOURCE device required when copying data from it onto a TARGET device. If the SOURCE device is identified by its VOLID, it must be either the only volume with that VOLID, or it must be the only VOLUME with that VOLID which is up (DVCUP), otherwise an error message will be issued. The whole VOLUME will normally be copied unless the operator has provided additional information that either identifies a cylinder range or a Data-Set-Name (DSN) contained on the source device that is to be copied.

*scyl-scyl*

The decimal start- and end-cylinder range where copying is to start and where it is to end on the source device. Cylinder is the smallest entity that can be specified for any SNAP command function. The highest (end) cylinder number must not exceed the device's primary number of cylinders and the start cylinder number must not be higher than the end cylinder number.

*scyl.ncyl*

The decimal start-cylinder where copying is to start on the source device and the number of cylinders (ncyl) that should be copied. Cylinder is the smallest entity that can be specified for any SNAP command function. The highest resulting cylinder number must not exceed the device's number of primary cylinders.

**DSN=** The data-set-name identifying the file on the source device, which must be a **non-VSAM** file, that the operator wants to be copied onto the target device. The file will be copied into the exact extent boundaries where it was located on the source device.

However, SAM (Sequential Access Method) files can be relocated (assuming that the level of the hardware support provides this function) to a different, single extent disk location on the target device. In this case, the **tcyl** operand must be supplied but the device must **not** be a **VM-partial minidisk**. The proper label information (single FORMAT-1 label) will be created and added to the target VTOC.

Processing multi-volume-files is the responsibility of the operator, such that the SNAP command should be repeated for all the source volumes containing file extents.

The number of cylinders to be copied is limited by the limits existing for the source device. Copying will only be performed if the appropriate extent boundaries on the TARGET device are available or have already expired, otherwise an error message will be provided. (Refer to the DDSR function in case the overlaid file should be deleted and released).

*target* The Device-ID (cuu) or the VOL1 label of the TARGET device is required when copying data to it from a SOURCE device.

The target device must be set DOWN (DVCDN command) prior to initiating the SNAP function, except the source and the target device are the same device (user is copying data from one location of a disk into another location on the same disk), or except a file (DSN=data-set-name) is being copied. If the TARGET device is identified by its VOLID, it must either be the only volume with that VOLID, or it must be the only VOLUME with that VOLID which is DOWN (DVCDN), otherwise an error message will be issued.

As many cylinders as allocated on the SOURCE device will be used for file copying onto the target device (DSN=data-set-name). Otherwise, as many cylinders as specified for the SOURCE device, or the whole SOURCE volume, will be copied onto the TARGET device.

If the specified cylinder range does not match the cylinder range that was given for the source device, relocation of data records will be assumed. This applies for **ESS** only, and providing the level of the hardware supports this function.

If the cylinder range does not match the cylinder range of the source device and the target device is a **VM partial-pack minidisk**, the command will be rejected. This is because VM uses virtual cylinder values for partial-pack minidisks, and the cylinder ranges must match for VM partial-pack minidisks. The source and the target device must be of the same type and must be in the same subsystem.

*tcyl* The decimal specification of where copying is to start on the target device. Cylinder is the smallest entity that can be specified for any SNAP command function. The target cyl specification (tcyl) added to the specified or calculated ncyl-1 value for the source device is the resulting target end cylinder address and it must not exceed the device's primary number of cylinders.

*VOL1=valid*

The VOL1 label that the TARGET device is to receive after the source volume has been copied. This operand is required if unique VOLIDs are to be maintained, otherwise the source and the target device would have the same VOL1 label after the copy function has completed. The VOL1 label specification for a target device will only be accepted when both, the cyl and the DSN= specification have been omitted (which means copying a full VOLUME).

### **NOCOPY**

Indicates that a physical copy of the source data (Volume, DSN, or cylinders) on the specified target is not required. This keyword is useful when creating a backup tape and a physical copy is not required. When the backup tape has been created, the target device is usually no longer

required. It can therefore be deleted (using DDSR) and the relation terminated. The NOCOPY relation exists until it is explicitly reset using the DDSR command.

However, you should be aware that the NOCOPY option does **not** imply that the target device will not be used. If the subsystem is running short of CACHE storage, it will use the TARGET device internally.

#### **NOPROMPT**

Prevents decision-type messages to be issued. Some messages require an operator reply before the specified function is going to be initiated. The specification of the NOPROMPT keyword will cause the system to bypass this decision-type message and will initiate the function without any additional notice.

#### **DDSR**

Delete Data Space Request (DDSR) is a command requesting an eventual ongoing SNAP command to be terminated before physical copying of the entities specified in that SNAP command has been completed and/or the associated File-Id, if any, to be deleted from the VTOC. DDSR, if specified for a unit without any additional operands for a device which is the target device of a SNAP ...NOCOPY relation, will cause this NOCOPY relation to be terminated for the specified device. Since such a target device does not represent a physical copy of its associated source device, it must **NOT BE USED** once the DDSR command has been completed.

This requires the associated volume to be re-initialized (ICKDSF) before using it as a regular data-pack again (assuming it is not going to be used as a SNAP target device in which case no initialization is required). VSE requires the volume to be down (DVCDN command) if the whole volume is to be deleted.

*unit* This is the Device-ID (cuu) or the VOL1 label of the device that should either be totally released, or, in case a data-set-name (DSN=data-set-name) identifies the device containing the File-Id of the file those label information is to be deleted from the VTOC.

**DSN=** This is the data-set-name, identifying the file on the specified unit, which must be a non-VSAM file, that the operator wants to be deleted. If the specified unit is in the UP (DVCUP) state, then the label information for this file will be deleted from the VTOC. If the device is down (DVCDN), the command will be rejected and an error message is provided. Processing multi-volume-files is the responsibility of the operator, such that the DDSR command must be repeated for all the volumes containing file extents.

#### **STATUS**

The IXFP STATUS function provides information about the current status and progress of ongoing or persisting FlashCopy relations. This function has no parameters. These devices must have been added during IPL.

### **Using IXFP SNAP with VM Minidisks**

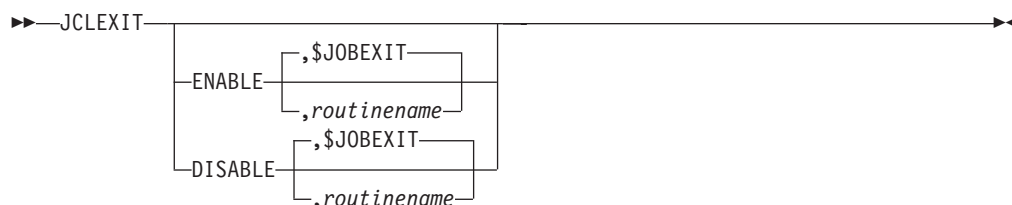
You have to consider that for minidisks which are using MDC (Mini Disk Caching) the MDC buffer must be flushed before performing a SNAP or DDSR function, otherwise data can be incomplete. The MDC problem is solved by VM APAR VM61486.

**Note:** Other host caching products (for example, Cache Magic) will have the same requirements.

## JCLEXIT (Multiple JCL Exits)

The JCLEXIT command activates or deactivates either a single JCL exit routine or (by default) all routines listed in the phase \$JOBEXIT. For details on JCL exit routines, see “Multiple Job Control Exit Routines” in the *z/VSE Guide to System Functions*.

### JCC Format



#### ENABLE[,routine name]

Indicates that the specified JCL exit routine is to be activated.

#### DISABLE[,routine name]

Indicates that the specified JCL exit routine is to be deactivated.

If you omit **routine name** (or if you specify \$JOBEXIT for it), all routines listed in \$JOBEXIT will be activated or deactivated. Depending on what is specified in \$JOBEXIT, this means activating or deactivating a single JCL exit routine or a list of JCL exit routines.

If no operand is specified in the JCLEXIT command, you get a report on SYSLOG about the status (enabled or disabled) of **all** JCL exit routines. The JCLEXIT command without an operand can be issued from any partition. With operands it can only be issued in the BG partition.

## JOB (Identify Job)

The JOB statement indicates the beginning of control information for a job.

### JCS Format

```
▶▶ // JOB jobname _____
                        |_____|
                        |accounting_information|
```

#### jobname

The name of the job. Must be one to eight alphanumeric characters (0-9, A-Z, #, \$, @) or slash (/), hyphen (-), or period (.). When a job is restarted, the jobname must be identical to that used when the checkpoint was taken. Any user comments can appear on the JOB statement following the jobname (through column 71). The time of day appears in columns 69–99 when the JOB statement is printed on SYSLST. The time of day is printed in columns 1-31 on the next line of SYSLOG.

In both cases the format is

```
DATE mm/dd/yyyy, CLOCK hh/mm/ss
```

**mm/dd/yyyy** can also appear in the format **dd/mm/yyyy**, if this was specified in the STDOPT command.

#### accounting information

If the job accounting interface has been specified during system installation, the 16 characters of user information are moved to the job accounting table. If accounting information is specified, it must be separated from the job name by a single blank. If the job accounting interface has not been specified during system generation, any information specified after the job name is ignored.

#### Notes:

1. If the JOB statement is omitted from the job stream, no duration is printed at end of job (when the /& statement is read).
2. The start time that the job control program displays is taken from the time-of-day clock (job step start time). The stop time for any given step is the start time for the next step.

The layout of the job accounting table is described under “Job Accounting Information” in the *z/VSE Guide to System Functions*.

## LFCB (Load Forms Control Buffer)

The LFCB command causes the system to load a buffer image, stored as a phase in the system sublibrary IJSYSRS.SYSLIB, into the forms control buffer (FCB) of the specified printer. The command can be used for any printer on which forms skip operations are controlled by an FCB.

For an IBM 4248 printer in native mode, however, the horizontal copy function cannot be activated or deactivated by the LFCB command.

If you have VSE/POWER in your system, use the \* \$\$ LST control statement with the FCB operand for this purpose. During the time the printer in question is printing the output of a program, this command should be used with extreme caution, as there is no way of predicting when the printer will be finished printing the output under control of the buffer image currently contained in the FCB.

For a printer in operation it is recommended that the operator uses this command if, for example, printing the output for a particular program started under control of the wrong FCB image and he is able to correct this by issuing the command.

### AR Format

▶—LFCB *cuu,phasename* —————▶  
                                   └──,FORMS=xxxx──┘ └──,NULMSG──┘

#### **cuu**

Specifies the device number of the printer whose FCB is to be loaded.

#### **phasename**

Specifies the name of the phase that contains the applicable buffer load image. For detailed information on the contents and format of this phase refer to "Buffer Load Phases" on page 357.

#### **FORMS=xxxx**

Specifies the installation-defined forms number *xxxx* of the paper that is to be used with the new FCB image. For *xxxx*, substitute from one to four alphameric characters. If the new FCB image requires a change of forms, this operand must be specified to ensure proper system operation.

#### **NULMSG**

Specifies that the printing of a buffer load verification message is to be suppressed. If NULMSG is specified, the system continues processing immediately after the FCB load operation has been completed, and the operator is unable to verify that the contents of the FCB match the forms to be used.

## LIBDEF (Define Sublibrary Chain)

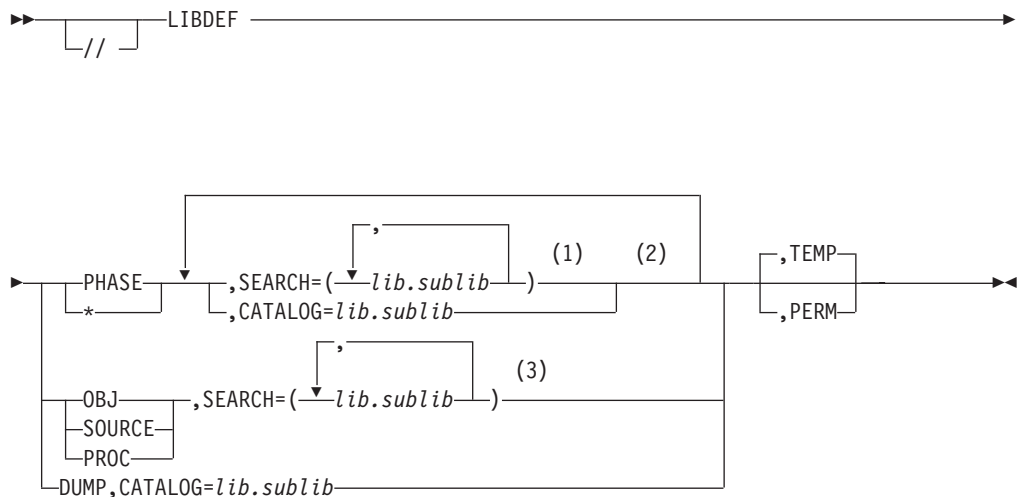
The LIBDEF statement defines which sublibraries are to be searched for members of a specified type or types, and, where appropriate, the sublibrary in which new phases or dumps are to be stored. The defined sequence is referred to as a search chain. Different chains can be defined for different member types, or a common chain can be established for all types except DUMP. The specified sublibraries are searched in the sequence as entered in the LIBDEF command or statement.

The system sublibrary IJSYSRS.SYSLIB is always added at a default position in the search chain, unless it is explicitly included at a different position in the chain. For details, see "Phase Chaining" on page 128.

### Notes:

1. The librarian program does not use the information given in LIBDEF statements to access sublibraries.
2. Continuation lines are allowed for this statement.

### JCC, JCS Format



### Notes:

1. Up to 32 sublibraries can be specified.
2. Duplicate keywords are not allowed.
3. Up to 32 sublibraries can be specified.

### PHASE

Defines a sublibrary chain to be used for loading or fetching program phases for execution. Only members of the type PHASE are searched for. The CATALOG operand specifies the library and sublibrary in which phases are to be cataloged by the linkage editor.

### OBJ

Defines a sublibrary chain to be used by the linkage editor when searching for object modules. Only members of the type OBJ are searched for. The CATALOG operand is not applicable.

### SOURCE

Defines a sublibrary chain to be used, for example by language translators,



when searching for one of the predefined "SOURCE" types (A-Z, 0-9, #, \$, @). The CATALOG operand is not applicable.

### PROC

Defines a sublibrary chain to be used by Job Control when searching for a procedure to be executed. Only members of the type PROC are searched for. The CATALOG operand is not applicable.

**Note:** If a LIBDEF PROC... or LIBDEF \*... statement is cataloged in a procedure, this procedure must fulfill the following criteria:

- It must reside in the system sublibrary IJSYSRS.SYSLIB;
- It must not be nested;
- It must not contain an EXEC PROC statement after the LIBDEF statement.

### DUMP

Defines a sublibrary to be used by the system when a dump is to be produced and the option SYSDUMP is in effect, or a CANCEL command with the SYSDUMP operand is issued. You must use the keyword CATALOG if you specify DUMP as the type operand.

- \* Indicates that the LIBDEF statement applies to all member types except DUMP and user types. That is, a common chain for all other member types is established. If the CATALOG operand is specified, it will apply for members of the type PHASE only. See Note under type PROC.

### SEARCH=lib.sublib

Is required if you specified OBJ, SOURCE or PROC in the type operand. With type PHASE, or \* you must specify SEARCH or CATALOG or both. The specified sublibraries will be searched in the sequence in which they are specified in this operand.

For all types of member except system phases, the system sublibrary IJSYSRS.SYSLIB is added at the end of the chain by default, unless you specify it explicitly at another position in the operand list. If the system sublibrary is added by default, then access control (if active) checks only the universal access rights for the system sublibrary. Any higher individual access right to this sublibrary is ignored.

For a LIBDEF statement with the type PHASE, the system directory list (SDL) may also be specified explicitly in the operand list. The default chaining sequence used when searching for phases depends on whether they are system phases or not. For details, see "Phase Chaining" on page 128.

**Note:** You may specify a list of up to 32 sublibraries in one search chain. The sublibrary names in the list must be separated by commas. If you specify only one sublibrary, it need not be enclosed in parentheses.

### CATALOG=lib.sublib

Is applicable for LIBDEF statements with the type PHASE, DUMP or \* only. It specifies the library/sublibrary into which the linkage editor or DUMP output is to be cataloged. There is no system default.

### TEMP | PERM

Specify the duration of the definition given in the statement. If you specify TEMP, the defined chain will be dropped:

- At end-of-job, or
- When overridden by a new LIBDEF..TEMP statement or command, or

- When overridden by a LIBDROP...TEMP statement or command.

If PERM is specified, the chain will remain valid until:

- The partition is deactivated by an UNBATCH command, or
- In the case of a dynamic partition, the partition is deallocated, or
- A LIBDROP...PERM statement or command is issued, or
- A new LIBDEF statement overrides the definition wholly or in part.

If you omit both of these operands, TEMP will be assumed by default.

If both a TEMP and a PERM LIBDEF statement have been issued for a given member type, the following rules apply:

- For SEARCH, the TEMP search chain is placed logically before the PERM chain.
- For CATALOG, the TEMP library definition is used. If a sublibrary protected by the access control function is specified in a **permanent** LIBDEF command or statement, this sublibrary **must** have a universal access right of connect or higher. For a **temporary** LIBDEF, the normal security checking is done. That is, the universal access right or the individual access right of the user who enters the command, whichever is the greater, is used.

### Phase Chaining

The search chain for phases includes the system directory list (SDL), and is different for “system” and “non-system” phases. In this context, “system phases” are phases which:

1. Have a name starting with a dollar sign (\$), or;
2. Are being loaded with the SYS=YES operand in the LOAD macro.

The search chains are:

- For “non-system” phases: SDL -- TEMP chain -- PERM chain -- IJSYSRS.SYSLIB.
- For “system” phases: SDL -- IJSYSRS.SYSLIB -- TEMP chain -- PERM chain.

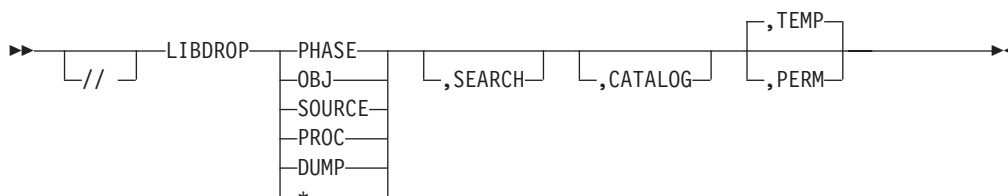
If you do not wish the SDL to be searched first, you must specify it at the appropriate position in the operand list of the temporary LIBDEF statement for phases. (SDL can be specified **only in a temporary** LIBDEF statement). You must not place IJSYSRS.SYSLIB or IJSYSR1.SYSLIB to IJSYSR9.SYSLIB before the SDL in a search chain.

For the access control aspects of the LIBDEF statement, see “Access Control for LIBDEF Statements” in the *z/VSE Guide to System Functions*.

## LIBDROP (Drop Sublibrary Chain)

The LIBDROP statement resets the library search and catalog definitions set up by one or more previous LIBDEF statements. Continuation lines are allowed for this statement.

### JCC, JCS Format



#### PHASE

Specifies that the search and catalog definitions for phases are to be reset.

#### OBJ

Specifies that the search definition for object modules is to be reset. The CATALOG operand is not accepted if OBJ is specified.

#### SOURCE

Specifies that the search definition for predefined "SOURCE" types (A-Z, 0-9, #, \$, @) is to be reset. The CATALOG operand is not accepted if SOURCE is specified.

#### PROC

Specifies that the search definition for members of type PROC is to be reset. The CATALOG operand is not accepted if PROC is specified.

**Note:** If a LIBDROP PROC ... or LIBDROP \*... statement is cataloged in a procedure, the following criteria apply to this procedure:

- It must reside in the system sublibrary IJSYSRS.SYSLIB
- It must not be nested
- It must not contain an EXEC PROC statement after the LIBDROP statement

#### DUMP

Specifies that the sublibrary definition for dump files is to be reset. If you have specified DUMP, the SEARCH operand is not accepted.

- \* Specifies all types except DUMP. If operands SEARCH and CATALOG are both specified, all previous PERM or TEMP definitions for search chains and catalog libraries will be dropped.

#### SEARCH

Specifies that only the search chain is to be dropped.

#### CATALOG

Specifies that only the sublibrary defined in the CATALOG operand of a previous LIBDEF statement is to be dropped.

**Note:** If neither SEARCH nor CATALOG is specified, both chains are dropped.

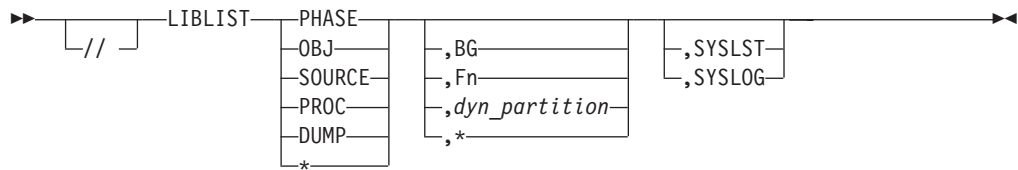
#### TEMP | PERM

Specify whether the temporary (TEMP) or permanent (PERM) definition is to be dropped. The system default is TEMP.

## LIBLIST (Query Sublibrary Chains)

The LIBLIST statement causes the library definitions set up with the LIBDEF statement to be displayed on SYSLOG or SYSLSST. Continuation lines are allowed for this statement.

### JCC, JCS Format



#### PHASE

Displays the search chain and the catalog library established by the LIBDEF PHASE statement.

#### OBJ

Displays the search chain established by the LIBDEF OBJ statement.

#### SOURCE

Displays the search chain established by the LIBDEF SOURCE statement.

#### PROC

Displays the search chain established by the LIBDEF PROC statement.

#### DUMP

Displays the catalog library established by the LIBDEF DUMP statement.

\* Specifies all types except DUMP, and causes the library definitions of all LIBDEF statements to be displayed.

#### BG | Fn | dyn\_partition | \*

Specifies the static or dynamic partition for which the current library definitions are to be displayed. \* means all static (not dynamic) partitions. If this operand is omitted, the output shows the library definitions for the partition in which the LIBLIST command is entered.

#### SYSLSST | SYSLOG

Specifies the output device to be used for displaying the library definitions. If this operand is omitted, SYSLSST will be used, unless the LIBLIST command was entered at SYSLOG, in which case the information will be displayed there.

## LIBSERV (Control IBM 3494 Tape Library Dataserver)

The LIBSERV command allows to pass requests to an IBM 3494 Tape Library Dataserver. These requests are communicated to the IBM 3494 Tape Library Dataserver in the following way:

- Using the TLS support, (IPL SYS ATL=TLS). For details, see *z/VSE Administration*.
- Using the VSE Guest Server (VGS) machine in VM, (IPL SYS ATL=VM). For details, see *z/VM V4R3.0 DFSMS/VM Function Level 221 Removable Media Services (SC24-6050-00)* .
- Using the VSE/ESA Library Control Device Driver for IBM 3494, (IPL SYS ATL=VSE).

The LIBSERV command communicates tape related information via an application programming interface (LIBSERV macro). Thus tape device handling can be achieved without any operator intervention. Volumes can be automatically retrieved from the tape library, queried, mounted, demounted after processing, and returned to the tape library.

The values to be specified in LIBSERV command parameters such as LIB and UNIT highly depend on definitions that have been made in your configuration files.

- For TLS support see TLSDEF.PROC residing in IJSYSRS.SYSLIB, or the sample job TLSDEF in VSE/ICCF library 59.
- For VGS support see the customization exit FSMRMVGC, and LIBCONFIG LIST on the A-disk of the VGS machine.
- For LCDD support see the LCDD startup job LCARUN in VSE/ICCF library 59.

### LIBSERV AQUERY

JCC, JCS Format:

```

▶▶ [//] LIBSERV AQUERY,VOL=volser

```

### LIBSERV CANCEL

AR Format:

```

▶▶ LIBSERV CANCEL,UNIT=cuu

```

### LIBSERV CMOUNT

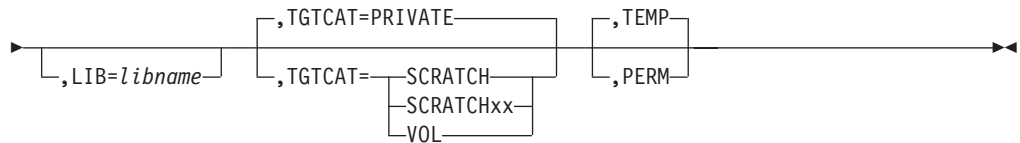
JCC, JCS Format:

```

▶▶ [//] LIBSERV CMOUNT,SRCCAT=
    [//] INSERT
    [//] SCRATCH
    [//] SCRATCHxx
    [//] ,UNIT=[//] cuu
    [//] SYSxxx

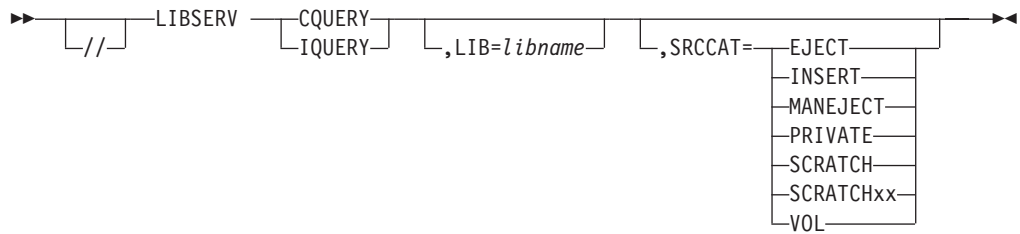
```

# LIBSERV



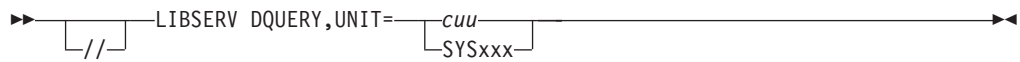
## LIBSERV CQUERY, IQQUERY

JCC, JCS Format:



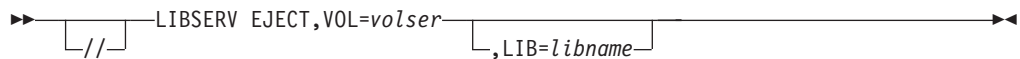
## LIBSERV DQUERY

JCC, JCS Format:



## LIBSERV EJECT

JCC, JCS Format:



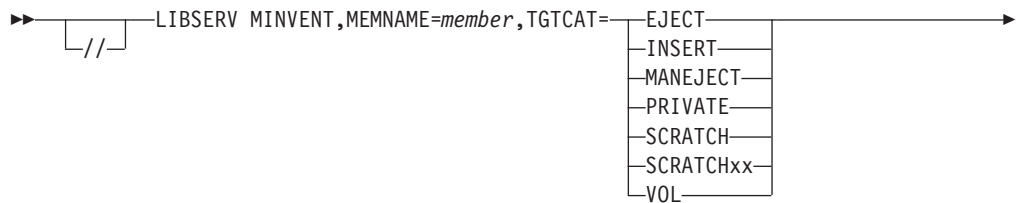
## LIBSERV LQUERY

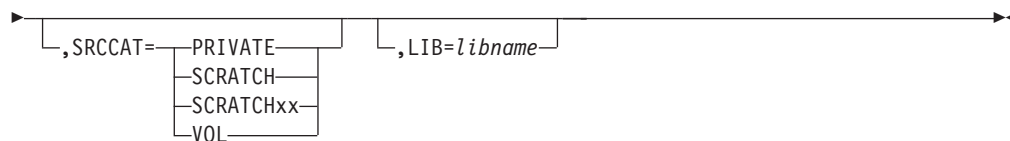
JCC, JCS Format:



## LIBSERV MINVENT

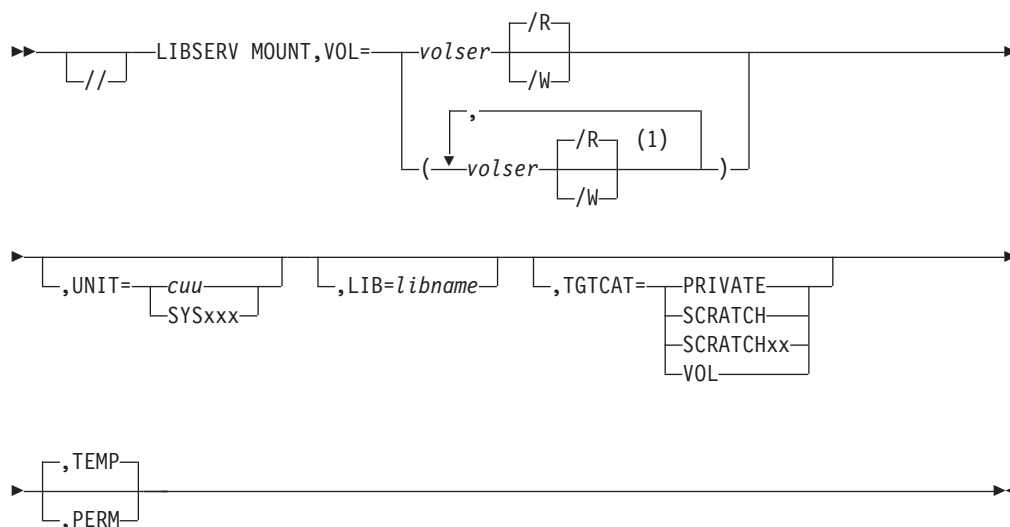
JCC, JCS Format:





## LIBSERV MOUNT

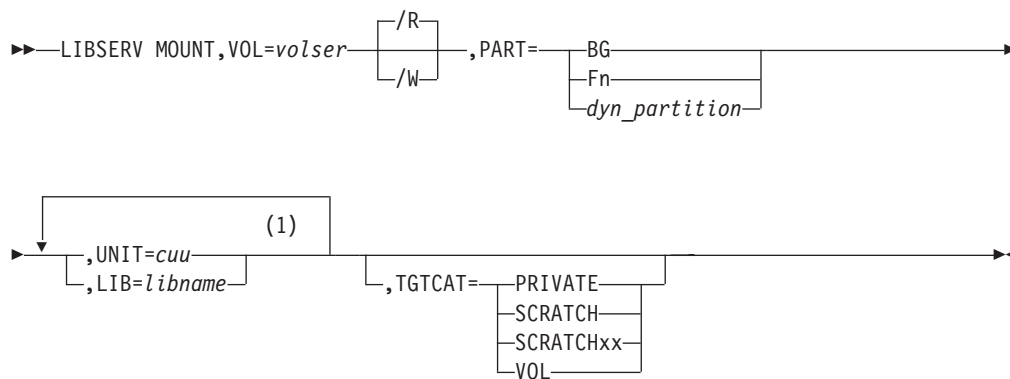
JCC, JCS Format:



### Notes:

1 Up to 10 volume serial numbers can be specified.

### AR Format:

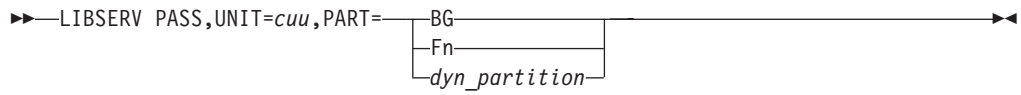


### Notes:

1 Each item can be specified only once.

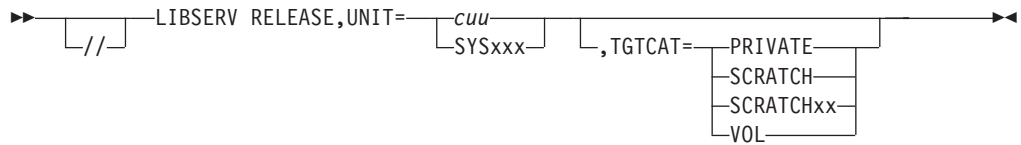
## LIBSERV PASS

AR Format:

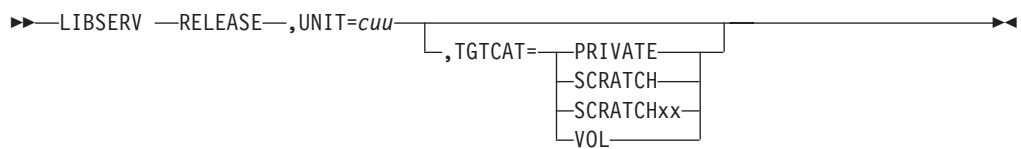


## LIBSERV RELEASE

JCC, JCS Format:

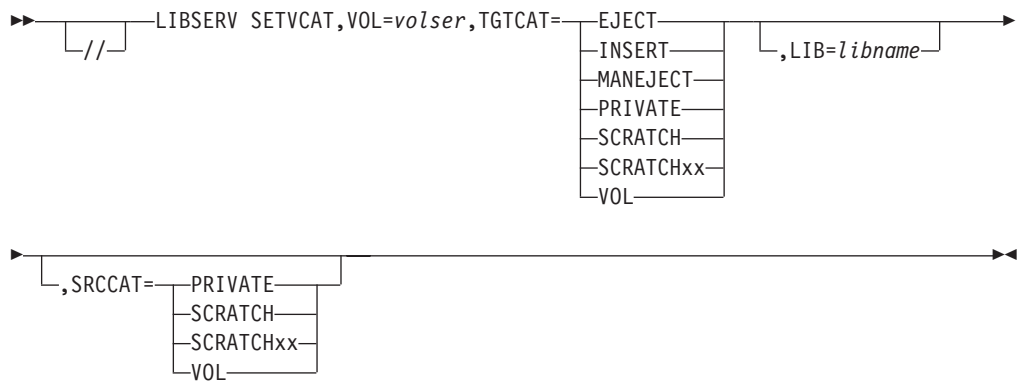


AR Format:



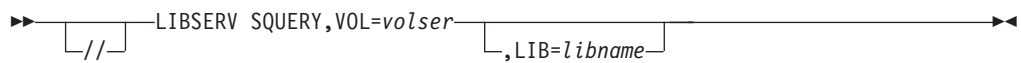
## LIBSERV SETVCAT

JCC, JCS Format:



## LIBSERV SQUERY

JCC, JCS Format:





## Explanation of Keywords

The required function must be the first keyword after LIBSERV. All other keywords can be specified in any arbitrary sequence. Continuation lines are accepted.

### AQUERY (Query Volume, All Libraries)

Is used to determine the library location of a specified volume. All libraries defined in the respective configuration file are queried, until the specified volume is found, and the library location is returned in a message, along with additional information on status and source category.

### CANCEL

In case a preceding CMOUNT or MOUNT operation has not yet been started, or it is in a hang condition, this can be canceled by using LIBSERV CANCEL. An ongoing CMOUNT or MOUNT request cannot be stopped, but the pointers and assignments will be reset.

### CMOUNT (Category Mount)

Is used to mount the next available volume from a specified source category (SRCCAT) on a library device at a specified virtual address (UNIT).

If a programmer logical unit SYSxxx is specified with the UNIT operand, then the library device is used, to which SYSxxx has been assigned to.

If the UNIT operand is omitted, the system searches for a free tape drive in the specified library.

After successful completion, this cuu is available for I/O in the partition in which LIBSERV was given.

A CMOUNT request is not possible if the cuu is already mounted or assigned in another partition. If a CMOUNT request is issued for a cuu that was already mounted, the cuu remains attached to the partition and the new MOUNT overwrites the old one. Between the first and the second CMOUNT request an MTC RUN command or statement is required to rewind and unload the tape. If the (second) CMOUNT fails, an implicit RELEASE request is issued against the cuu.

The category for the mounted volume can be changed by specifying the TGTCAT operand. If the TGTCAT operand is omitted, the category for the mounted volume is changed to PRIVATE. The specified device (UNIT) must be located in the specified library (LIB). If the library name is omitted, then the default library is assumed.

### CQUERY (Count Query)

Is used to count the number of volumes currently assigned to a specified category in a specified library. If the source category is omitted, then the entire library inventory is counted. If the library name is omitted, then the default library is used.

### DQUERY (Drive Query)

Is used to determine the status of a specified tape drive. The following information is returned in a message:

- The status of the tape drive.
- The volser of the mounted volume (if any).
- The category of the mounted volume (if any).

### EJECT

Removes a volume from a specified library or from the default library if the library name is not specified in the request. If the request is successful, the volume is moved to the output station for removal from the library by the operator.

If the cartridge to be EJECTed is currently mounted, message 1YH8t is issued with RET.CODE=08 and REASON=3336 (volume in use). If a LIBSERV EJECT request is given right after a LIBSERV RELEASE request, the volume may still be in use because rewind or unload processing has not yet finished. Therefore, if a job needs to eject a volume right after it has been processed, the following job control is recommended:

```
// LIBSERV MOUNT,VOL=xxxxxx,UNIT=yyy  Mount cartridge
... tape processing ..Work with cartridge
// LIBSERV RELEASE,UNIT=yyy          Release cartridge
// EXEC IESWAIT,PARM='30'           Wait 30 sec for completion Rewind/Unload
// LIBSERV EJECT,VOL=xxxxxx         Move cartridge to output station
```

### **IQUERY (Inventory Query)**

Is used to request inventory data on volumes currently assigned to a specified category, in a specified library. If the source category is omitted, then inventory data for the entire library is provided. If the library name is omitted, then the default library is used.

The inventory data is placed in a librarian-managed file in lib.sublib as specified in the respective configuration file. An Inventory Query request fails if the configuration file does not specify a lib.sublib to contain the inventory data or if lib.sublib was not defined via LIBR.

### **LIB=libname**

Specifies the name of the library in which the operation is requested. libname must have been defined in the respective configuration file. libname can be one to eight characters. Printable characters except blanks, equal signs, parentheses and commas are allowed. If the operand is omitted, the default library is used.

### **LQUERY (Library Query)**

Is used to determine the operational status of a library. If the library name is omitted, the default library is queried.

### **MEMNAME=member**

Specifies the name (up to 8 characters) of a librarian-managed file, containing the list of volumes to be processed by MINVENT requests. The corresponding lib.sublib must have been defined in the respective configuration file.

### **MINVENT (Manage Inventory)**

Is similar to the SETVCAT request, but processes a list of volumes specified in MEMNAME rather than a single volume (as SETVCAT does). All volumes contained in the specified list and located in the specified library are assigned to the specified target category. If an optional source category is specified, the category designation for a volume is changed only if the volume is currently assigned to the specified source category. The category designation for a volume is changed only if it is located in the specified library. If the library name is omitted, then the default library is assumed. Each volume's record in MEMNAME is updated to show whether the category change was processed successfully or not.

### **MOUNT**

Is used to mount a volume with a specified external label (VOL) on a library device at a specified virtual address (UNIT).

If a programmer logical unit SYSxxx is specified with the UNIT operand (only allowed in the JCC, JCS format), then the library device is used, to which SYSxxx has been assigned to.

If the UNIT operand is omitted, the system searches for a free tape drive in the specified library.

After successful completion, this cuu is available for I/O in the partition in which or for which LIBSERV was given.

A MOUNT request is not possible if the cuu is already mounted or assigned in another partition. If a MOUNT request is issued for a cuu that was already mounted, the cuu remains attached to the partition and the new MOUNT overwrites the old one. Between the first and the second MOUNT request an MTC RUN command or statement is required to rewind and unload the tape. If the (second) MOUNT fails, an implicit RELEASE request is issued against the cuu.

The category for the mounted volume can be optionally changed by specifying the TGTCAT operand.

Both the volume (VOL) and the device (UNIT) must be located in the specified library (LIB). If the library name is omitted, then the default library is assumed.

#### **PART=BG | Fn | dyn\_partition**

Indicates the 2-character partition identifier to which a cuu is to be MOUNTed or PASSed. This is only needed for the AR LIBSERV command. The JCL LIBSERV command only works for the partition in which it is specified.

If the partition identifier specified corresponds to a partition currently not active then an error message is displayed.

#### **PASS**

Causes the cuu specified in the UNIT operand to be 'transferred' to the partition specified in the PART operand, provided the specified cuu is mounted to a partition and this partition has no assignments to the cuu. After completion, the cuu can only be assigned and used by the partition to which it was passed.

#### **/R | /W**

Specifies the access mode for this volume. If present, it must immediately follow the volume serial number.

**/R** Indicates that the volume is mounted in read-only access (which is the default).

#### **/W**

Indicates that the volume is mounted in read/write access.

#### **PERM**

Indicates a permanent mount request: the cuu can be released only by an explicit RELEASE request. Only permanent MOUNTs can be given from AR.

#### **RELEASE**

Specifies that the previously mounted library device specified in the UNIT operand is to be taken away from the partition to which it was mounted.

If a programmer logical unit SYSxxx was specified with the UNIT operand, then the physical tape unit, to which SYSxxx had been assigned, is RELEASED from the partition to which it was mounted.

A rewind unload is issued for the specified device and the actual volume on the device will be returned to the tape library.

The category for the volume previously mounted on the specified UNIT can be optionally changed by specifying the TGTCAT operand.

Unless all assignments are released, the cuu can only be mounted again where it was mounted before. Note that an unsuccessful MOUNT request also works like RELEASE.

#### **SETVCAT (SET Volume CAtegory)**

Is used to assign a volume located in a specified library to a specified target category. If an optional source category is specified, the category designation for the volume is changed only if the volume is currently assigned to the specified source category. The category designation for the volume is changed only if it is located in the specified library. If the library name is omitted, then the default library is assumed.

#### **SQUERY (Query Volume, Single Library)**

Is used to determine if a specified volume is located in the specified or default library. If a specific library name is not specified as input in the request, the default library is queried. If found, additional information on status and source category is returned in a message.

#### **SRCCAT=sourcecat**

Specifies the name of the source category (up to 10 characters).

#### **TEMP**

Indicates a temporary mount request: the cuu is to be released automatically at end-of-job if still mounted. This is the default for a job control MOUNT or CMOUNT request.

#### **TGTCAT=targetcat**

Specifies the name of the target category (up to 10 characters).

#### **UNIT=cuu | UNIT=SYSxxx**

cuu indicates the channel and unit number of the library device, which must have been defined in the respective configuration file. If SYSxxx is specified together with the UNIT operand (only allowed in the JCC, JCS format), then the systems recalls the physical unit SYSxxx has been assigned to, and processes the request for this physical unit.

If the UNIT operand is omitted in a MOUNT or CMOUNT request, then the system searches for a free tape drive associated with the tape library whose name is specified in the LIB operand. If the LIB operand has been omitted, too (only allowed in the JCC, JCS format), then a free tape drive associated with the default library is searched.

#### **VOL=volser**

Indicates the external volume serial number as known by the tape library. The volume serial number can be one to six characters long. If you use less than six characters, the field is padded on the right with blanks. If access mode is also specified (see above), then the field must either be exactly six bytes long or enclosed in quotes. Note that a volume serial number cannot end with /R or /W unless it is enclosed in quotes (see access mode above). Also, it must not contain any blanks, equal signs, parentheses, or commas. In the JCL command or statement (not in the AR command) you can also specify a list of volume serial numbers (enclosed in parentheses). In this case the next volume is automatically mounted after the previous one has been processed (for example, with a multi-volume file).

Figure 5 on page 139 shows a sample scenario of how the LIBSERV command can be used to simplify and automate tape handling.

```

=> 4 LIBSERV LQUERY                                1
    F4-0004 1YL6I  LIBRARY QUERY , LIB : TAPELIB1  STATUS : 0000
=> 4 LIBSERV SQUERY,VOL=VSER02                    2
    F4-0004 1YL2I  VOLUME FOUND IN LIB : TAPELIB1 SRCCAT : SCRATCH05 STATUS :
    0000
=> 4 LIBSERV MOUNT,VOL=VSER02                      3
    F4-0004
=> 4 ASSGN SYS005,TPA,VOL=VSER02                  4
    F4-0004 1T20I  SYS005 HAS BEEN ASSIGNED TO X'AA0' (PERM)
=> 4 LIBSERV DQUERY,UNIT=AA0                      5
    F4-0004 1YL5I  DEVICE QUERY,VOLUME : VSER02   SRCCAT : SCRATCH05 STATUS :
    0000
.
.
=> 4 LIBSERV RELEASE,UNIT=SYS005,TGTCAT=PRIVATE  7
    F4-0004
=> 4 ASSGN SYS005,UA                              8
    F4-0004
=> 4 LIBSERV SQUERY,VOL=VSER02                    9
    F4-0004 1YL2I  VOLUME FOUND IN LIB : TAPELIB1 SRCCAT : PRIVATE STATUS :
    0000

```

Figure 5. JCC Scenario for LIBSERV Command

#### Explanation:

- 1** An LQUERY request is given. Because the LIB operand has been omitted, status information for the default library (TAPELIB1) is returned in message 1YL6I.
- 2** An SQUERY request is given for volume VSER02. Because the LIB operand has been omitted, the default library (TAPELIB1) is queried. Information on volume status and source category is returned in message 1YL2I.
- 3** A MOUNT request is given for volume VSER02. Because both UNIT and LIB operand have been omitted, the system
  - checks, whether volume VSER02 is contained in the default library (TAPELIB1).
  - searches for a free tape drive associated with the default library (TAPELIB1). If there was no free tape drive available, message 1YK0t would be displayed.
  - mounts volume VSER02 and establishes mount-ownership for partition F4.
- 4** The programmer logical unit SYS005 is to be assigned to a device with device-type code TPA. Since VSER02 is specified with the VOL operand, the system checks all TPA devices to see if volume VSER02 is mounted. Note, that in the context of LIBSERV the VOL operand denotes the *external* volume serial number as known by the tape library. In the context of ASSGN, however, the VOL operand denotes the volume serial number as written on the cartridge. In the sample scenario, both volume identifiers match, so the system finds TPA device AA0. Now the logical unit SYS005 is permanently assigned to the tape drive AA0 in the F4 partition. The F4 partition is prepared for tape I/O.
- 5** A DQUERY request is given for tape drive AA0. Status information, the mounted volume, and its source category are returned in message 1YL5I.

## LIBSERV

- 6** Programs may work with tape unit AA0.
- 7** A RELEASE request is given. No *cuu* but the programmer logical unit SYS005 is specified with the UNIT operand. The system recalls tape device AA0 as the physical unit SYS005 had been assigned to and annuls F4's mount-ownership for the tape. A rewind unload is issued for device AA0, and volume VSER02 is returned to the tape library.

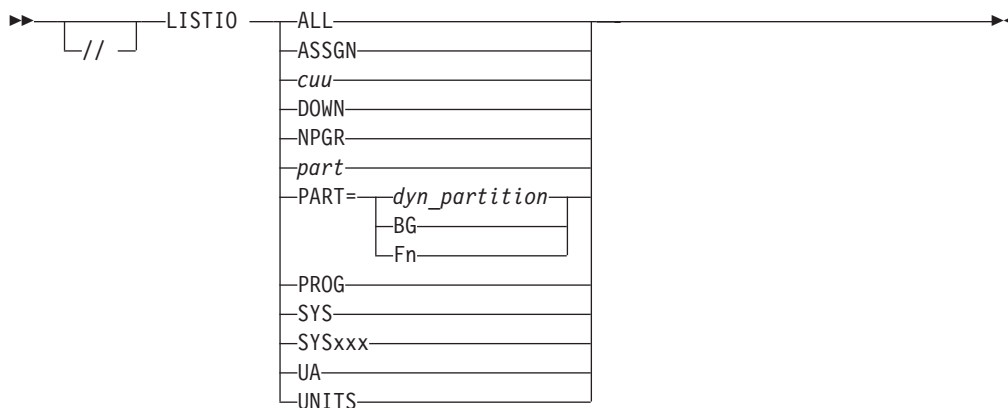
Since SYS005 is still assigned to AA0 in the F4 partition, no other partition can issue a LIBSERV MOUNT request for AA0 (message 1YBnt would be the result).

- 8** The logical unit SYS005 is unassigned. From now on other partitions may issue MOUNT requests for tape device AA0.
- 9** Another SQUERY request is given for volume VSER02. This time source category PRIVATE is returned in message 1YL2I, because the RELEASE request in **7** specified TGTCAT=PRIVATE.

## LISTIO (Query I/O Assignments)

The LISTIO command or statement causes the system to print a listing of I/O assignments. The listing appears on SYSLOG (for the command format) or SYSLST (for the statement format). If SYSLST is not assigned, the // LISTIO statement is ignored.

### JCC, JCS Format



**listtype** can be one of the following:

#### ALL

Lists the physical units assigned to all logical units of all static (not dynamic) partitions.

#### ASSGN

Lists the physical units assigned to all system and programmer logical units of the partition from which the command is issued. Unassigned units are **not** listed.

#### cuu

Lists the logical units assigned to the specified physical unit. See Note 2.

#### DOWN

Lists all physical units specified as inoperative.

#### NPGR

Lists the number of programmer logical units allocated to each partition.

#### part

Lists the physical units assigned to all logical units of the specified static or dynamic partition: BG, Fn, or a dynamic partition (except UA). For a dynamic partition named 'UA', use the format PART=UA. See Note 1.

#### PART=dyn\_partition | BG | Fn

PART=dyn\_partition indicates that you want to list the devices of a *dynamic* partition. You must use this format if your dynamic partition is named 'UA'.

PART=BG and PART=Fn have the same effect as the specifications BG and Fn, respectively (see above, under 'part'). Both notations can be used interchangeably.

#### PROG

Lists the physical units assigned to all programmer logical units of the partition from which the command is issued. See Note 1.

## LISTIO

### **SYS**

Lists the physical units assigned to all system logical units of the partition from which the command is issued. See Note 1.

### **SYSxxx**

Lists the physical units assigned to the specified logical unit of the partition from which the command is issued (invalid for SYSOUT and SYSIN).

### **UA**

Lists all physical units not currently assigned to a logical unit.

### **UNITS**

Lists the logical units assigned to all physical units of all (static and dynamic) partitions. See Note 2.

### **Notes:**

1. Unassigned logical units are listed as UA.
2. Unassigned or inoperative physical units are listed as UA or DOWN, respectively.

In addition, physical units for which a mount request for a IBM 3494 Tape Library Dataserver has been initiated, are listed as MNTP (mount pending). Also, physical units for which a mount request for a IBM 3494 Tape Library Dataserver has been completed but not yet released, are listed as MNTPC (mount complete).



## LOG (Log JC Statements)

The LOG command or statement causes the system to log all job control commands and statements occurring within the scope of the LOG.

The scope of the LOG depends on which form is used, as follows:

- For the attention routine command, it depends on the specified operand (see below). Columns 1 to 72 of all logged commands and statements are written to **SYSLOG**. LOG remains effective until an attention routine NOLOG command or a job control NOLOG command (for the issuing partition) is given.
- The job control command LOG affects the partition in which it is issued. Columns 1 to 72 of all logged commands and statements are written to **SYSLOG**. LOG remains effective until a job control NOLOG command for the same partition, or an attention routine NOLOG command for all partitions, is given.
- The job control statement // LOG affects the partition in which it is issued. Columns 1 to 80 of all logged commands and statements, including the // LOG statement itself, are written to **SYSLST**. // LOG is effective until end-of-job, or until a // NOLOG statement occurs in the same job.

**Note:** The // LOG statement has the same effect as the // OPTION LOG statement. The two statements are interchangeable, and each can be reset by either the // NOLOG or the // OPTION NOLOG statement.

To have job control statements and commands of a given job logged only on SYSLST, use the job control NOLOG command in the appropriate partition and the // LOG statement in the job.

The // LOG statement can be used to print comments on SYSLST.

The LOG command or statement suppresses OPTION ACANCEL. That is, it prevents jobs being canceled because of an unsuccessful ASSGN or LIBDEF attempt.

### AR Format



### JCC Format



### JCS Format



### ALL

Lists the job control statements and commands of all static and dynamic partitions.

## LOG

### **BG | Fn | dyn\_partition**

Lists the job control statements and commands of the specified static or dynamic partition.

### **class**

Lists the job control statements and commands of the specified dynamic class.

If no operand is specified in the attention routine command, LOG causes the job control statements and commands of all static partitions to be listed.

## LUCB (Load Universal Character-Set Buffer)

The LUCB command causes the system to load the buffer image, contained in the named phase, into the universal character set buffer (UCB) of the specified printer. The printer must be ready or in operation. The command can be used for any printer equipped with the UCS feature, except the 1403U, and the 4248 printer in native mode (that is, added with the device type 4248 at IPL).

For the 4248, use the attention routine command BANDID.

While a printer is printing the output of a program, this command should be used with caution. There is no way of knowing when the printer has finished printing the output under control of the buffer image currently contained in the UCB.

For a printer in operation it is recommended that the operator use this command if, for example, printing the output for a particular program started under control of the wrong UCB image and the operator is able to correct this condition by issuing the command.

### AR Format

▶▶—LUCB *cuu,phasename* —┬──┬──┬──┬──┬──┬──▶

└──,FOLD──┘ └──,NOCHK──┘ └──,TRAIN=xxxxxx──┘ └──,NULMSG──┘

#### **cuu**

Specifies the device number of the printer whose UCB is to be loaded.

#### **phasename**

Specifies the name of the phase which contains the applicable buffer load image. For detailed information on the contents and format of this phase refer to "Buffer Load Phases" on page 357.

#### **FOLD**

Causes lowercase characters to be printed as uppercase characters.

#### **NOCHK**

Causes data checks resulting from mismatches between print-line characters and the UCB to be suppressed.

#### **TRAIN=xxxxxx**

Indicates that the print train identified by xxxxxx is to be mounted on the printer. The system inserts this operand in an action message. The train identification xxxxxx may be from one to six characters in length. If a new train (or chain) must be installed, this operand is required to ensure proper system operation.

#### **NULMSG**

Specifies that the printing of a buffer load verification message is to be suppressed. If NULMSG is specified, the system continues normal processing immediately after the UCB load operation has been completed and the operator is unable to verify that the contents of the UCB match the print train (or chain) mounted on the printer.

## MAP

### MAP (Display Storage Layout)

The MAP command produces, on SYSLOG, a map of all storage areas in the system, together with their sizes and starting addresses.

#### JCC, AR Format



If the MAP command is given without an operand, MAP VIRTUAL is assumed.

Note that in the following output examples the sizes shown under V-SIZE and GETVIS are not necessarily identical with the **current** values. This is, for example, the case when the sizes are overridden by an EXEC...SIZE= statement.

#### MAP VIRTUAL

The following output example shows the current virtual storage layout, that is, how the IPL VSIZE (the TOTAL value) is distributed. It provides information for the supervisor, the SVA and the static partitions, plus summary information for dynamic partitions and data spaces.

```
MAP
AR 0015 SPACE AREA      V-SIZE  GETVIS  V-ADDR  UNUSED NAME
AR 0015 S SUP           656K    0        0      $$A$SUPX
AR 0015 S SVA-24       1648K   1664K   A4000   128K
AR 0015 0 BG V         1280K   4864K   400000  45056K
AR 0015 1 F1 V          788K    812K   400000   0K POWSTART
AR 0015 2 F2 V         2048K   49152K  400000   0K CICSICCF
AR 0015 3 F3 V          600K    5544K  400000   0K VTAMSTRT
AR 0015 4 F4 V         2048K   18432K  400000   0K
AR 0015 5 F5 V          768K    256K   400000   0K
AR 0015 6 F6 V          256K    256K   400000   0K
AR 0015 7 F7 V         1024K   11264K  400000   0K
AR 0015 8 F8 V         2048K   49152K  400000   0K
AR 0015 9 F9 V          256K    256K   400000   0K
AR 0015 A FA V          256K    256K   400000   0K
AR 0015 B FB V          256K    256K   400000   0K SECSERV
AR 0015 S SVA-31      3956K   7308K  3600000
AR 0015 DYN-PA         0K
AR 0015 DSPACE        4448K
AR 0015 SYSTEM        1024K
AR 0015 AVAIL         83168K
AR 0015 TOTAL        256000K  <----'
AR 0015 1140I READY
```

The maximum amount of private space a partition may use below the 16 MB line can be calculated by subtracting all shared areas (below 16 MB) from 16 MB:

16 MB - Supervisor V-SIZE - SVA-24 (V-SIZE+GETVIS+UNUSED)  
(assuming the complete SVA-31 to be above the 16 MB line)

## MAP REAL

The following output example shows the current real storage layout, that is, how the total real storage available to z/VSE (shown by the TOTAL value) is currently allocated. It provides detailed information for the supervisor, the PFIX values for the system and the static partitions, and the ALLOC R values for the static partitions. It also provides summary information on the PFIX values of all dynamic partitions.

MAP REAL does not refer to the actual usage of real storage.

```

MAP REAL
AR 0015      AREA      R-SIZE R-ADDR      PFIX (BELOW)      PFIX (ABOVE)
AR 0015      AREA      R-SIZE R-ADDR      ACTUAL  LIMIT      ACTUAL  LIMIT
AR 0015      SUP        592K      0
AR 0015      SYS-24          1800K 14576K
AR 0015      BG V          0K      0K      0K      0K
AR 0015      F1 V          68K     152K     0K      0K
AR 0015      F2 V          28K     144K     0K      0K
AR 0015      F3 V          68K     424K     0K      0K
AR 0015      F4 V          0K      0K      0K      0K
AR 0015      F5 V          0K      0K      0K      0K
AR 0015      F6 V          0K      0K      0K      0K
AR 0015      F7 V          0K      0K      0K      0K
AR 0015      F8 V          0K      0K      0K      0K
AR 0015      F9 V          0K      0K      0K      0K
AR 0015      FA V          0K      0K      0K      0K
AR 0015      FB V          0K      0K      0K      0K
AR 0015      SYS-31          28K     28K
AR 0015      DYN-PA          0K      0K      0K      0K
AR 0015      AVAIL        64K
AR 0015      SYSTEM      404K
AR 0015      TOTAL      16384K      <----'      <----'
AR 0015
AR 0015      AVAILABLE FOR SETPFIX:      12776K      0K
AR 0015

```

## MAP CLASS=ALL

This format shows the address limits for **all** currently defined dynamic partition classes in the system. The values related to a class are shown in one line. The line contains the class character, virtual storage size, GETVIS size, space-GETVIS size, virtual start address of the class, number of active partitions in the class, and the maximum number of partitions in the class.

```

MAP CLASS=ALL
AR 015 CLASS V-SIZE  GETVIS SP-GETV V-ADDR A-PART M-PART
AR 015 G    200K    696K  128K    0    0    2
AR 015 H    700K    100K  224K 338000 2    20
AR 015 I    848K    48K   128K    0    0    5
AR 015 J    300K    596K  128K    0    0    32
AR 015 K    800K    96K   128K    0    0    32
AR 015 L   1024K    896K  128K    0    0    10
AR 015 M   3500K    340K  256K    0    0    32
AR 015 N    512K    384K  128K    0    0    32
AR 015 O    700K    196K  128K    0    0    1
AR 015 C    600K    296K  128K    0    0    12
AR 015 1140I  READY

```

**Note:** Class priorities are not included in this output; they can be displayed via the PRTY command.

## MAP

### MAP CLASS=class

This format shows the address limits for a specific dynamic class. In the example shown below, the header line and the line for class H have the same content as shown under CLASS=ALL above. The next line is a header line for the specified class (H).

The example shows a MAP output for class H with two jobs:

```
MAP CLASS=H
AR 015 CLASS V-SIZE   GETVIS SP-GETV V-ADDR A-PART M-PART
AR 015   H       700K   100K   224K 338000   2    20
AR 015
AR 015 PART PWR-JOB  JOBNUMBER  PFIX(BELOW)    PFIX(ABOVE)
AR 015           ACTUAL   LIMIT  ACTUAL   LIMIT
AR 015   H1 PAUSEH           20      0K   120K   0K     0K
AR 015   H2 PAUSEI           21      0K    0K    0K     0K
AR 015 1140I  READY
```

### MAP Partition

This format displays detailed information for the specified partition, like jobname and phasename, virtual storage and real storage allocation.

#### Example for a Dynamic Partition:

```
MAP H1
AR 015 PARTITION: H1           SPACE-GETVIS.....: 224K  ADDR: 300000
AR 015 CLASS.....: H           ALLOC (VIRTUAL)...: 800K  ADDR: 338000
AR 015 STATUS...: VIRTUAL      SIZE.....: 700K
AR 015 POWER-JOB: PAUSEH      EXEC-SIZE.....: 700K
AR 015 JOBNUMBER: 20          GETVIS.....: 100K
AR 015 JOBNAME...: PAUSEH     EXEC-GETVIS...: 100K  ADDR: 3E7000
AR 015 PHASE.....: LIBR
AR 015                               PFIX(BELOW)-LIMIT: 120K
AR 015                               -ACTUAL: 0K
AR 015                               PFIX(ABOVE)-LIMIT: 0K
AR 015                               -ACTUAL: 0K
AR 015 1140I  READY
```

#### Example for a Static Partition (without ALLOC R):

```
MAP BG
AR 015 PARTITION: BG           SPACE-GETVIS.....: (N/A)
AR 015 SPACE.....: 0           ALLOC (VIRTUAL)...: 704K  ADDR: 300000
AR 015 STATUS...: VIRTUAL      SIZE.....: 600K
AR 015 POWER-JOB: PAUSEBG
AR 015 JOBNUMBER: 43          GETVIS.....: 104K  ADDR: 396000
AR 015 JOBNAME...: PAUSEBG
AR 015 PHASE.....:
AR 015                               PFIX(BELOW)-LIMIT: 704K
AR 015                               -ACTUAL: 0K
AR 015                               PFIX(ABOVE)-LIMIT: 0K
AR 015                               -ACTUAL: 0K
AR 015 1140I  READY
```

**Example For a Static Partition (With ALLOC R and EXEC REAL):**

```

MAP F3
AR 015 PARTITION: F3      SPACE-GETVIS.....: (N/A)
AR 015 SPACE.....: 3      ALLOC (VIRTUAL)...: 320K  ADDR: 300000
AR 015 STATUS....: REAL   SIZE.....: 272K
AR 015 POWER-JOB: PAUSEF3
AR 015 JOBNUMBER: 6      GETVIS.....: 48K  ADDR: 344000
AR 015 JOBNAME...: PAUSEF3
AR 015 PHASE.....: LIBR
AR 015                ALLOC (REAL).....: 100K  ADDR: EE2000
AR 015                EXEC-SIZE.....: 100K
AR 015                PFIX(ABOVE)-LIMIT: 0K
AR 015                -ACTUAL: 0K
AR 015 1140I  READY

```

**MAP SVA**

```

MAP SVA
AR 015 SPACE AREA      V-SIZE  GETVIS  V-ADDR
AR 015  S  SVA-24      876K   1364K  900000
AR 015  S  SVA-31      136K   1912K  1300000
AR 015
AR 015          AREA      R-SIZE  R-ADDR  PFIX(BELOW)  PFIX(ABOVE)
AR 015                ACTUAL  LIMIT  ACTUAL  LIMIT
AR 015          SYS-24                528K  14264K
AR 015          SYS-31                120K   888K
AR 015 1140I  READY

```

**Explanation of Keywords****A-PART**

The number of active partitions in the class.

**AREA**

This column identifies to which storage area the data in the respective line refers:

```

AVAIL = Amount of storage still available for allocation
BG, Fn = Static partitions
R      = Job step in partition is running in real mode
V      = Job step in partition is running in virtual mode
S      = Partition has been stopped (by a STOP command)
I      = Partition inactive
DSPACE = Data space (summary information)
DYN-PA = Dynamic partition (summary information)
SUP     = Supervisor area
SVA     = Shared virtual area
SVA-24 = SVA adjacent to the supervisor (the 24-bit SVA)
SVA-31 = SVA at the high end of the storage (the 31-bit SVA)
SYSTEM = Amount of storage used by the system outside the supervisor,
        the SVA (virtual), and the PFIX areas (real).
TOTAL  = Maximum possible allocation in the system.

```

**CLASS**

Dynamic partition class character.

**EXEC-GETVIS**

Partition allocation minus EXEC-SIZE.

**EXEC-SIZE**

Size of program area within the specified partition as defined with the EXEC SIZE job control statement or command.

**GETVIS**

Size of permanent GETVIS space in the area. This is partition size minus V-SIZE (ALLOC specification minus SIZE command specification). If no SIZE command has been given for the partition, GETVIS is 48K plus virtual storage above the 16 MB line.

In the SVA-24 line, GETVIS includes the V-pool area and a 108K byte work area for static partitions.

**JOBNAME**

The name of the VSE job currently running in the specified partition.

**JOBNUMBER**

The job number of the VSE/POWER job in the specified class.

**M-PART**

The maximum number of partitions per class.

**NAME**

The name of the supervisor (SUP line), or the job currently running in the respective partition. If the job accounting file has become full, this column shows the job name JOB ACCT instead of the name of the current job. When no JOB statement has been entered in the partition, this field contains blanks.

**PART**

The partition-id of the dynamic partition within the class (H in the example).

**PFIX(ABOVE)**

Shows, per partition and for the system area above the 16MB line (SYS-31), the amount of storage that is actually PFIXed or that may be PFIXED (as a maximum limit) in page frames **above** the 16MB line.

For partitions, the maximum has been defined by the SETPFIX command. For the system area (SYS-31), the maximum is a system-defined value. This value always includes the amount of storage that is shown under AVAILABLE FOR SETPFIX (see MAP REAL), since this storage is automatically made available to the system.

The amount of storage shown as AVAILABLE FOR SETPFIX is available for setting a partition's PFIX limit via the SETPFIX statement.

**PFIX(BELOW)**

Shows, per partition and for the system area below the 16MB line (SYS-24), the amount of storage that is actually PFIXed or that may be PFIXED (as a maximum limit) in page frames **below** the 16MB line.

For partitions, the maximum has been defined by the SETPFIX command. For the system area (SYS-24), the maximum is a system-defined value. This value always includes the amount of storage that is shown under AVAILABLE FOR SETPFIX (see MAP REAL), since this storage is automatically made available to the system.

The amount of storage shown as AVAILABLE FOR SETPFIX is available for setting a partition's PFIX limit via the SETPFIX statement.



**PHASE**

The name of the phase currently executing in the specified partition.

**PWR-JOB / POWER-JOB**

The name of the VSE/POWER job in the specified class.

**R-ADDR**

The starting address (in address space R) of the real storage in the area.

**R-SIZE**

The amount of real storage in the area:

- In the SUP line, the size of the supervisor.
- In the static partition lines, the size of the real partition (ALLOC R).
- In the AVAILable line, the amount of real storage which is still available for real storage allocation (ALLOC R).
- In the SYSTEM line, the amount of real storage which is reserved for page frame tables and the minimum page pool.
- In the TOTAL line, the amount of real storage available to z/VSE. (For a VM user, it shows the amount specified via DEF STOR.)

**SPACE**

The space-id (0 through B) of the address space in which the respective area is located. S indicates the shared address space. The space-id R does not occur; real allocations appear in the R-SIZE and R-ADDR columns of MAP REAL.

A \* after the space-id indicates that the partition in this address space has been allocated by using 'multiple-partition allocation' (see ALLOC command).

**SPACE-GETVIS and SP-GETV**

For dynamic partitions only: Shows the size of the dynamic class space GETVIS area.

**UNUSED**

Shows, in the last (or only) line of an address space, the amount of non-allocated storage in the address space. This storage may be used to allocate further partitions or to increase the existing partitions within this address space.

Please note that address spaces which have been created with "single-partition allocation" (see ALLOC command) normally have an unused space of 0K. For such an address space, the difference between PASIZE and the initial partition allocation size is not usable for partition allocation and thus is not included in the UNUSED value. Only if the size of such a partition has been decreased, is the difference between the initial and the new allocation size shown as unused space and may be used to increase this partition if needed lateron.

In the SVA-24 line, UNUSED equals the following:

```
specified SPSIZE + area required for 1Mb rounding below the
line - allocation of shared partitions.
```

All components and also UNUSED are multiples of 64K.

**V-ADDR**

The starting address of the area in virtual address space. When referring to part of an area, for example in a DUMP, DSPLY or ALTER command, use the space-ID in the SPACE column.

### V-SIZE

The amount of virtual storage in the area.

- In the SUP line, the size of the supervisor plus the size of the SDAID area.
- For the static partitions, this is the value specified in the SIZE command for the partition. If no SIZE command has been used, V-SIZE is the partition size, as specified in the ALLOC command, minus 48K and minus all virtual storage above the 16MB line.
- For the DYN-PA line, the total amount of virtual storage currently allocated to dynamic partitions.
- For the DSPACE line, the total amount of virtual storage currently allocated to data spaces.
- For the SYSTEM line, the amount of virtual storage used by system routines outside the supervisor and the SVA, that is, system routines that run in separate address spaces.
- For the AVAILable line, the amount of virtual storage which is available for partition and data space allocation.
- For the TOTAL line, the IPL VSIZE value.

## MSECS (Change or Query Time Slice)

The MSECS command displays or changes the time slice for partition balancing.

### JCC Format

►► MSECS *n* ◄◄

### AR Format

►► MSECS ◄  
└─*n*─┘ ◄◄

- n** Specifies the base on which the new time slice in milliseconds is calculated. This is the period of processor time after which the priorities of partitions in a partition balancing group are inspected and potentially rearranged. *n* must be an integer from 100 to 10000. The default is about 1000 milliseconds. Values of less than 100 milliseconds would lead to higher overhead without better partition balancing.

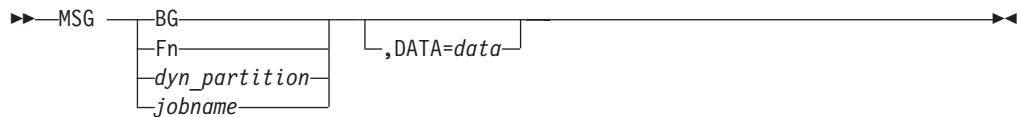
If you use the attention routine command without the operand, the system displays the current time slice in milliseconds.

The attention routine command MSECS can be entered at any time, but the job control command can be used only within an ASI procedure. Here you must code a valid value for *n*.

## MSG (Communicate With Program)

The MSG command transfers control to an operator communications (OC) routine for which linkage has been established with a STXIT macro. The command can also be used to pass data to the OC exit.

### AR Format



#### **BG | Fn | dyn\_partition**

Indicates the partition-id of the partition (static or dynamic) for which the OC exit is to be dispatched.

#### **jobname**

Indicates the job name of the job for which the OC exit is to be dispatched. *jobname* can be up to 8 characters and must be unique.

#### **DATA=data**

Contains the data to be passed to the OC exit routine.

If the program in the specified partition has not established operator communication linkage, a message is printed on SYSLOG informing the operator of this condition.

Depending on the OC exit option MSGPARM=YES, command text, if any, is saved in system GETVIS storage (24- or 31-bit, depending on the OC exit AMODE option), and fields ARCONSID, ARCONSNM and ARCART are copied, together with a pointer to the command text, into corresponding fields of the OC exit save area. In this case, the maximum length of command text, specified with the MSG command and passed directly to the OC exit, may be up to 126 characters.

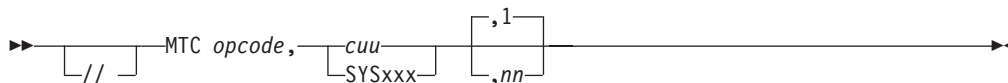
If system GETVIS storage is not available, the message INSUFFICIENT SVA STORAGE is generated.

If the OC exit was defined with the MSGDATA option, and more than 64 bytes of input data are specified, the message INPUT DATA TOO LONG is generated.

## MTC (Magnetic Tape Control)

The MTC command or statement controls magnetic tape operations.

### JCC, JCS Format



### AR Format



#### opcode

Specifies the operation to be performed as explained in Figure 6.

#### SYSxxx

Specifies the logical unit to which the tape is assigned.

#### cuu

Specifies the device number.

**nn** Is a decimal number from 1 through 99 that indicates the number of times the specified operation is to be performed. The default is 1.

Opcode	Meaning	Possible Use
BSF	Backspace file	Backspace one file so tape is positioned for reading the tapemark preceding the file backspaced.
BSR	Backspace Record	Backspace record.
DSE	Data Security Erase	(See Note)
ERG	Erase Gap	Erase gap.
FSF	Forward Space File	Used when restarting a program. The tape is positioned beyond tapemark following the file spaced over.
FSR	Forward Space Record	Locate a specific record within a file.
RUN	Rewind and Unload	Rewind and unload a tape on a specific unit.
REW	Rewind	Rewind a tape on a specific unit.
WTM	Write Tape Mark	Write a tapemark on an output file.
<p>Note: Data security erase (3400-series only). This command erases a tape from the point at which the tape is positioned when the operation is initiated up to the end-of-tape reflective marker. If data is written after the end-of-tape reflective marker, the data must be erased with [/] MTC ERG,SYSxxx.</p>		

Figure 6. Operation Codes for MTC Statement

## MTC

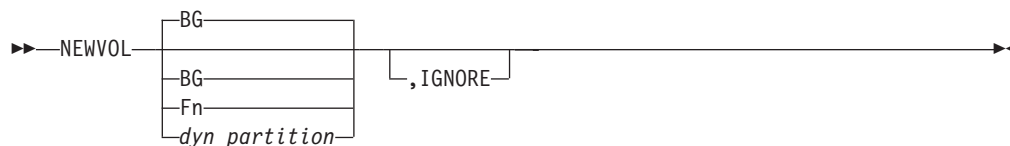
**Note:** If the MTC DSE command is issued when the tape is at load point, the contents of the tape, including the volume label, are erased completely. In such a case the tape must be reinitialized or a tapemark must be written on it before it can be used again.

The partition that issued the [//] MTC DSE command is placed in the wait state until the end-of-tape reflective marker is reached.

## NEWVOL (Alter Volume Assignment)

If an assignment specifying VOL= was given for a disk or tape unit and the system cannot find the requested volume on that unit, then the system prints message 1T50A on SYSLOG, requesting the operator to mount the desired volume. The partition enters the wait state. The operator can now either mount the proper volume, make the device ready, and issue the NEWVOL attention command to indicate that processing may continue with the new volume, or - if the volume cannot be mounted - he can cancel the mount request by specifying the IGNORE operand.

### AR Format



### BG | Fn | dyn\_partition

Indicates the static or dynamic partition for which the new volume was mounted. If no operand is specified, BG is assumed. If the specified partition is not waiting for a volume to be mounted, an error message is printed on SYSLOG.

### IGNORE

Specifies that the mount request is to be ignored. This causes message 1T40D to be displayed, after which you can either give a new assignment, or cancel the job, or enter any other command.

## NOLOG (Suppress JC Logging)

The NOLOG command or statement stops the listing of job control commands and statements (except ALLOC, DVCUP, HOLD, IGNORE, JOB, MAP, PAUSE, PRTY, SIZE, STOP, UNBATCH, /\*, /&, and /+) that occur within the scope of the NOLOG. The scope depends on the form of the NOLOG, as follows:

- For the attention routine command NOLOG, it depends on the specified operand (see below). It is effective until an attention routine LOG command or a job control LOG command (for the issuing partition only) is given.
- The job control command NOLOG affects only the partition in which it is issued.
- The job control statement // NOLOG affects only the partition in which it is issued. It overrides any previous // LOG statement for the remainder of the job. The // NOLOG statement itself is written to SYSLST.

**Note:** The // NOLOG statement has the same effect as the // OPTION NOLOG statement. These statements are interchangeable. Each can be used to reset either a // LOG or a // OPTION LOG statement.

### AR Format



### JCC Format



### JCS Format



#### ALL

Stops listing the job control statements and commands of all static and dynamic partitions.

#### BG | Fn | dyn\_partition

Stops listing the job control statements and commands of the specified static or dynamic partition.

#### class

Stops listing the job control statements and commands of the specified dynamic class.

If no operand is specified in the attention routine command, NOLOG stops listing the job control statements and commands of all static partitions.



## NPGR (Number of Programmer Logical Units)

The NPGR command defines the number of programmer logical units which may be allocated in a given static partition. The command is not allowed for a dynamic partition.

### JCC Format



#### Notes:

- 1 Each option may be specified only once.

#### BG,Fn

Specify the partition for which you wish the system to allocate the given number (m) of programmer logical units.

The operand BG can only be specified in the BG partition itself, and only when no other partition has been started since IPL. The operand Fn can be specified in any static partition, but only before partition Fn itself is started for the first time after IPL.

- m** A decimal integer from 10 through 255. The total number of logical units for all partitions must not, however, exceed the number specified in the NPGR parameter at supervisor generation.

40 programmer logical units (SYS000 to SYS039) are available for each partition by default. The programmer logical unit names used **must** be in the range SYS000 to SYSnnn, where nnn = m - 1. The current number of programmer logical units allocated to each partition can be displayed with the LISTIO NPGR command.

## OFFLINE

### OFFLINE (Simulate DEVICE OR CHPID NOT READY)

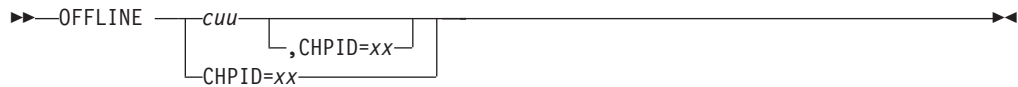
The OFFLINE command is used to force a device, a single channel path to a device, or a whole channel path into an inoperative state. The inoperative state is to be viewed as a LOGICAL-NOT-OPERATIONAL state rather than a PHYSICAL-NOT-OPERATIONAL state. Logical NOT-OPERATIONAL in this sense means that VSE will prevent I/O operations from being initiated via a certain channel path to either a single or to multiple devices. An appropriate message (OP31A) will be issued instead.

The System Operator Console (SYSLOG), the System Resident Device (SYSRES), and the SDAID output device (if applicable) cannot be OFFLINED at any time. At least one single path must always remain operational.

When issued for a TAPE device, VSE will first initiate an UNLOAD operation. The OFFLINED device will then be UNGROUPED thus giving other CPUs or the VM operating system the ability to GROUP and subsequently access that device via the same channel path(s).

The device will be set inoperative if there remains no further path to access the device.

#### AR Format



#### **cuu**

Identifies the device and/or the path that is to be set offline.

#### **CHPID=xx**

Identifies the id of the channel path that is to be set offline. xx is a hexadecimal number from 00 to FF.

Assignments remain unaffected by this command.

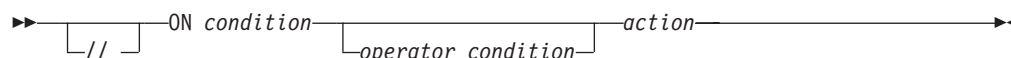
## ON (Set Global Condition)

The ON statement is a global conditional function. During execution of a job in which an ON statement occurs, the specified condition is tested at the end of each job step following the ON statement. If the condition is true, the specified action is taken, otherwise processing continues with the next statement.

For handling of conditional job control in VSE/POWER controlled jobs, refer to *VSE/POWER Administration and Operation*, chapter "Interaction with z/VSE Conditional Job Control Language".

The ON statement is accepted only within a job.

### JCC, JCS Format



#### Condition

Specifies the condition under which the action specified in the ON statement is to be taken. It may be expressed in one of the following forms:

\$RC comparator n  
\$CANCEL  
\$ABEND

where:

#### \$RC

Specifies the return code of the preceding step.

#### comparator

Specifies the comparison to be done. This can be one of the following six possibilities:

Comparison:	Specified as:
Equal to	= or EQ
Not equal to	≠ or NE
Greater than	> or GT
Less than	< or LT
Greater or equal	>= or GE
Less or equal	<= or LE

**n** Specifies a decimal integer from 0 to 4095, which is to be used for comparison with the return code.

#### \$CANCEL

Specifies that the action is to be taken if the CANCEL command is given for the job.

**Note:** If the job is canceled with the operand FORCE in the CANCEL command, the action specified in the ON statement is **not** carried out.

#### \$ABEND

Specifies that the action is to be taken if the step terminates abnormally, for example by causing a program check or issuing a CANCEL ALL macro.

#### operator

Specifies a logical operator which connects two conditions. The valid

## ON

specifications are: OR, |, AND, &. Each condition has the format described above. The specified action is taken at End-of-Job Step when:

- The conditions are connected by OR or |, and one or both of them is true;
- The conditions are connected by AND or &, and both of them are true.

The logical operators (OR, |, AND, &) must be preceded and followed by a blank character.

### action

Specifies the action to be taken if the specified condition is true at End-of-Job Step. It may be expressed in one of the following forms:

```
GOTO label  
CONT[INUE]
```

where:

#### GOTO label

Specifies that processing is to continue at the specified label statement. For rules on the use of this operand, see the explanation of the GOTO statement on page 113.

#### CONTINUE

Specifies that processing should continue if the specified condition is true. CONTINUE is not valid with the conditions \$CANCEL or \$ABEND.

Whenever a job starts, the following default ON conditions are in effect:

```
ON $RC<16 CONTINUE  
ON $RC>=16 GOTO $EOJ  
ON $ABEND GOTO $EOJ  
ON $CANCEL GOTO $EOJ
```

ON conditions remain in effect up to End-of-Job if they are specified outside of a procedure; if they are specified in a procedure, they are in effect up to the end of that procedure. If two or more ON statements at the same level specify the same condition, the action specified in the last one is carried out. For example, if your job specifies

```
ON $RC > 4 GOTO LABEL1  
ON $RC > 8 GOTO LABEL2
```

and a job step ends with return code 16, then the

```
GOTO LABEL2
```

will be executed. If you are using nested procedures, JC will check the condition of an ON statement on the level at which it is specified, and on all lower levels.

If you specify GOTO label as the action to be taken, the target label statement specified must be on the same level as the ON condition GOTO label statement, that is, either both in the same procedure, or both outside a procedure (on JC level 0).

If an ON condition occurs in a called (lower) procedure, the target of an associated GOTO will be searched for only in the procedure (or JC level 0) where the ON statement was specified, starting after the // EXEC PROC statement which called the procedure in which the condition was raised.

If a CANCEL command with the FORCE operand is given, or a job terminates due to a job control statement error, no ON conditions are checked: the rest of the job is skipped, provided the default option NOSCANCEL is in effect.

You can trap any job control statement error with an ON \$CANCEL condition, if you specify both SCANCEL and JCANCEL.

For an example of the use of the ON statement, see Figure 19 on page 262.

## ONLINE (Simulate DEVICE OR CHPID READY)

The ONLINE command is used to simulate a *not ready* to *ready* transition for the specified device or for all devices that are attached to the specified channel path (CHPID).

The devices will be set to the OPERATIONAL state if no conditions exist which would inhibit this (path or device has been QUIESCED). The option FORCE, if specified, will FORCE the specified device or the specified path to be set OPERATIONAL unconditionally.

When issued for an IBM 3480, 3490(E) or 3590 device, the ONLINE command causes the specified device (if any) to be dedicated to the issuing CPU. If the device is already 'dedicated' to another CPU, an appropriate message will be provided, and the device remains inaccessible unless the 'OWNING' CPU operator issues an OFFLINE command.

Onlining a device will cause the AVR process (Automatic Volume Recognition) to be initiated and the VCTE (Volume-Characteristic-Table-Entries) to be updated properly. The HOLD option, if specified for a device, will cause the device to not automatically be released (it remains 'dedicated' to this CPU) at EOJ time when no more assignments exist for this device.

### AR Format



#### cuu

Identifies either the device or the CHPID, in hexadecimal, for which the ready status is to be simulated.

#### CHPID=xx

Enables the operator to try setting the channel path indicated by xx in the operational state. xx is a hexadecimal number from 00 to FF.

#### FORCE

Enables the operator to **force** the device or CHPID in the operational state unconditionally.

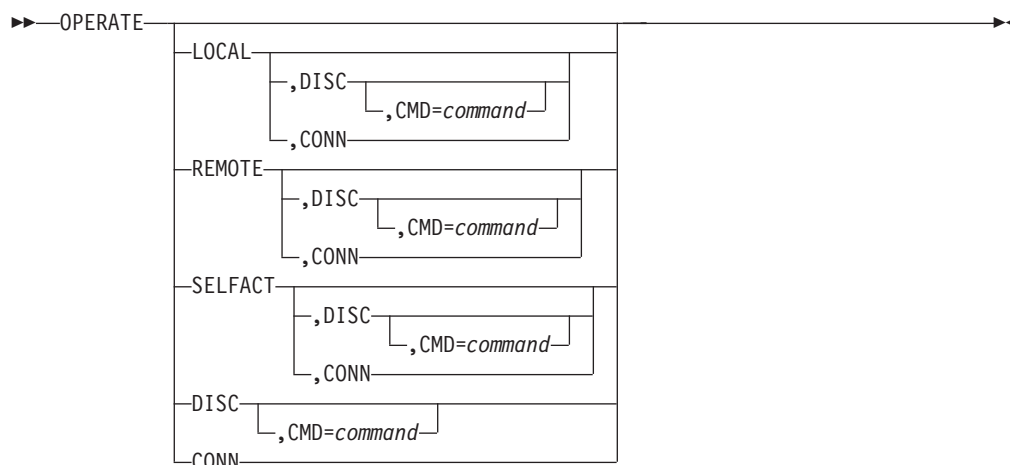
#### HOLD

Indicates that the device must not automatically be 'unassigned' from this CPU when no more VSE assignments exist. It can only be 'unassigned' by means of the OFFLINE command.

## OPERATE (Query or Alter Mode of Operation and Console State)

The OPERATE command allows the system operator to query or change the current system operating mode and the state of the system console.

### AR Format



The OPERATE command without any operand displays the current mode of operation (LOCAL, REMOTE, or SELFACT) and console state (DISC or CONN).

### LOCAL

Indicates that a local VSE system operator is attending and controlling the system. This is the normal mode of operation with a dedicated 3270 system console or with the integrated console. LOCAL is the default mode of operation for an attended system.

### REMOTE

Indicates that no local system operator is available and that the system is being controlled from another master console, possibly at a remote location. This operand is only valid in an unattended node environment (UNATT=YES was specified in the SYS command at IPL time). It is the default mode of operation in an unattended node environment once it has been customized properly. In this mode, input from both a 3270 system console (when defined) and from the integrated console is inhibited. When a message with reply cannot be routed to any console other than the system console, an automatic re-IPL is initiated.

### SELFACT

Indicates that no VSE system operator is available at all and that the system will be self-acting in case of special errors and events being encountered. For example, the system will:

- Re-IPL in case of hard waits
- Cancel I/O requests for ERP decision messages 0P00D - 0P69D
- Cancel the issuing task if a message requiring a reply cannot be routed to any active console other than the system console.

This mode of operation is primarily recommended for jobs that run for a long time and do not require any operator intervention. It is only valid in an attended node environment.

## OPERATE

Switching from one operation mode to another within the limitations stated above does **not** impact the console state (CONN or DISC) as described below:

### CONN

Indicates that messages can be routed to a 3270 system console or to the integrated console, even if the operating mode is REMOTE or SELFACT, preventing any replies to be entered from there. This is the initial state after the system is IPL-ed.

### DISC

Indicates that no messages are to be routed to a system console. This option is accepted only when the hardcopy file is open.

In this state, a 3270 system console will no longer be accessed as such by VSE and can therefore be used for other purposes (TP access methods). When the console is no longer used, it can be restored as operator console by means of the OPERATE CONN command or via an attention interrupt (for example, by hitting ENTER), with the same result as OPERATE CONN. Under VM, a DIALED 3270 console is reset and must be DIALED again before returning to the connected state.

Message traffic to the integrated console is also suppressed in the disconnected state and can only be resumed via OPERATE CONN.

The SYSLOG assignment cannot be changed in the disconnected state.

### CMD=command

Specifies a command that is to be issued as soon as the disconnected state is entered, for example, **VARY NET,ACT,ID=conid**, if VTAM is to take over the terminal.



## OPTION (Set Temporary JC Options)

The OPTION statement specifies one or more job control options which temporarily override the system defaults. These may have been changed by the STDOPT command. You may use the QUERY STDOPT command to find out which permanent job control options are currently active. You can also use QUERY OPTION to find out which actual job control options are currently active.

### JCS Format



The options specified in the OPTION statement remain active until a contrary option is encountered or until a JOB or a /& control statement is read. In the latter case, the options are reset to the system default values or those established by the STDOPT command.

Each option of the OPTION statement is processed separately. If processing of one option fails, the **following** options are **not** processed. The options specified **before** the failing option remain active.

If you enter conflicting options in one OPTION statement, the last one overrides the first.

**Note:** The PL/I VSE compiler does not recognize the options ERRS/NOERRS, LIST/NOLIST, LISTX/NOLISTX, RLD/NORLD, SYM/NOSYM, TERM/NOTERM, XREF/SXREF/NOXREF, or 48C/60C. The corresponding PL/I options of the \*PROCESS statement must be used instead.

The options, which can appear in any order, are as follows:

#### ACANCEL

Indicates that the job must be canceled (instead of awaiting operator intervention) if an ASSGN or LIBDEF command fails. This may happen because of an undefined device, invalid device status, unassignable unit, or conflicting I/O assignments.

The ACANCEL option is suppressed when either the LOG command has been issued by the operator or a LOG statement for the partition is effective.

#### NOACANCEL

Suppresses the ACANCEL option. The system awaits operator intervention in the case of an unsuccessful assignment.

#### ACL

Indicates that in case of multi-volume files and automatic cartridge loading being active on the actual device, the access method will process all tapes (cartridges) on the actual device first and then follow the alternate chain. (Normal ACL processing.)

#### NOACL

Specifies that in case of multi-volume files, the access method will follow the alternate chain, independent of whether automatic cartridge loading is active on the device or not.

## OPTION

### ALIGN

The assembler aligns constants and data areas on proper boundaries and checks the alignment of addresses used in machine instructions.

### NOALIGN

Suppresses the ALIGN option.

### CATAL

A phase or program is permanently cataloged in a library at the completion of a link-edit run. CATAL also sets the LINK option. (See also Note 1 on page 174.)

### CLASSTD=class

This option clears (overwrites) the class standard label group of the specified dynamic class. All DLBL and TLBL statements submitted after this point are written into the class standard label group of the specified dynamic class. Class must be a single class only.

OPTION CLASSTD remains in effect until one of the following occurs:

1. End of job or job step.
2. OPTION USRLABEL or STDLABEL or PARSTD is specified.

OPTION CLASSTD=class immediately clears the class standard label group of a dynamic class. See Note 5 on page 174.

### CLASSTD=(class,ADD)

All label information submitted after this option will be stored into the class standard label group of the specified dynamic class, without overwriting the information that already exists in that label group. The specified class must be a single class only. See Note 5 on page 174.

### CLASSTD=(class,DELETE)

This option must be followed by the filename(s) of the DLBL or TLBL statement(s) to be deleted from the class standard label group of the specified dynamic class. The last (or only) filename must be followed by /\*. After the /\*, the option CLASSTD=ADD becomes effective. CLASSTD=(class,DELETE) must be specified as the **last** option of the OPTION statement. The specified class must be a single class only. DELETE may be abbreviated by DEL. See Note 5 on page 174.

### DECK

Language translators produce object modules on SYSPCH. If LINK is specified, the DECK option is accepted by the PL/I, VS/FORTRAN, and DOS/VS COBOL compilers, and the assembler.

### NODECK

Suppresses the DECK option.

### DSPDUMP

Specifies that a data space dump is to be taken in case of an abnormal program end (and if OPTION DUMP or PARTDUMP is active). A data space is dumped

- If the failing program is in access-register mode when the abnormal program end occurs,
- **and** if the failing program has access to the data space (that is, the data space has an entry in the DU-AL or in the PASN-AL for the failing program),
- **and** if an access register contains the ALET (access list entry token) of the data space.

An area of at least 4K of storage on either side of the address(es) pointed to by the matching general register(s) is dumped. However, if the size of the data space does not exceed 128K of storage, the dump routine dumps the whole data space.

#### **NODSPDUMP**

Specifies that no data space dump is to be taken. This is the default.

Specify only one of the following options DUMP, PARTDUMP, or NODUMP in one OPTION statement:

#### **DUMP**

Dumps the registers, supervisor area, partition, the used part of the system GETVIS area, the SVA phase in error (if the error occurred in the SVA), and the phase load list. The dump is taken in the case of an abnormal program end (such as program check). (See also notes 3 and 4 on page 174).

#### **PARTDUMP**

Dumps selected areas of storage. For detailed information, refer to "Part 1. Dumps of Virtual Storage" in *z/VSE Diagnosis Tools*.

The dump is taken in the case of an abnormal program end.

#### **NODUMP**

Suppresses the DUMP or PARTDUMP option.

#### **ERRS**

The FORTRAN, DOS/VS COBOL, and PL/I compilers summarize all errors in the source program on SYSLST.

#### **NOERRS**

Suppresses the ERRS option.

#### **IGNLOCK**

Causes all possible locks to be ignored. A library member is then treated as if it had not been locked. Any librarian function in the job will delete, rename or update members even if they are locked, or will delete libraries/sublibraries or rename sublibraries even if they contain locked members. The renamed or updated member will be unlocked. Any LOCK or UNLOCK command will be ignored. In all cases, corresponding messages will be issued.

#### **NOIGNLOCK**

Suppresses the IGNLOCK option, and is the system default.

#### **JCANCEL**

Indicates that the system should skip to End-of-Job (instead of waiting for operator intervention) if a job control error occurs. See also SCANCEL.

#### **NOJCANCEL**

Suppresses the JCANCEL option, and is the system default. The system waits for operator intervention if a job control error occurs.

**Note:** If NOJCANCEL is specified, an AR LOG command causes certain I-type messages to be changed into D-type messages, thus allowing a corrected operator response to be entered. A NOLOG command changes the messages back to I-type.

#### **LINK**

Indicates that the object module is to be link-edited. When the LINK option is used it must always precede an EXEC LNKEDT statement in the input stream. Only a single phase can be link-edited. (See also Note 1.)

## OPTION

### NOLINK

Suppresses the LINK option.

### LIST

Language translators write the source module listing. The assembler also writes the hexadecimal object module listing, and the assembler and FORTRAN write a summary of all errors in the source program. All are written on SYSLST.

### NOLIST

Suppresses the LIST option. This option overrides the printing of the external symbol dictionary, relocation list dictionary (RLD), and cross-reference (XREF/SXREF) lists.

### LISTX

The COBOL compiler produces a PROCEDURE DIVISION map on SYSLST. The PL/I and FORTRAN compilers produce the object modules on SYSLST.

### NOLISTX

Suppresses the LISTX option.

### LOG

Lists columns 1-80 of all control statements and commands on SYSLST. Control statements and commands are not listed until a LOG option is encountered.

If LOG is the only option you want to specify, consider using the // LOG statement, which has the same effect and is shorter.

### NOLOG

Suppresses the listing of all valid control statements and commands on SYSLST until a LOG option is encountered. If SYSLST is assigned, invalid statements and commands are listed.

If NOLOG is the only option you want to specify, consider using the // NOLOG statement, which has the same effect and is shorter.

### LOGSRC

This option takes effect only when the option LOG is already in effect. It causes job control statements which contain symbolic parameters to be printed twice: once in source form (as coded), once with substituted symbolic parameters (as processed by job control.)

When the options LOGSRC and LOG are in effect, all statements skipped during execution are written to SYSLST. The reason for skipping them is indicated by the following character strings in columns 82 to 98:

\*\*\*/. **labelnam**\*\*\*

Skipped because of a GOTO statement.

\*\*\*\*\*

Skipped because of an IF statement.

\*\*\*/. **\$EOJ** \*\*\*

Skipped because the job was canceled.

### NOLOGSRC

Suppresses the LOGSRC option. If the option LOG is in effect, job control statements are printed once, showing the substitution of any symbolic parameters.

There is no corresponding option for the STDOPT statement. The system default is NOLOGSRC.

**PARSTD**

This option overwrites the partition's permanent label group. All DLBL and TLBL statements submitted after this point are written into the partition's permanent label group, to be available to all subsequent jobs in that partition until another PARSTD option without operand or with =DELETE is submitted.

OPTION PARSTD remains in effect until one of the following occurs:

1. End of job or job step.
2. OPTION USRLABEL or STDLABEL or CLASSTD is specified.

OPTION PARSTD immediately clears the partition's permanent label group. See Note 2 on page 174.

**PARSTD=ADD**

All label information submitted after this option will be stored into the partition's permanent label group without overwriting the information that already exists in that label group.

**PARSTD=(Fn,ADD)**

This option has the same function as PARSTD=ADD, except that it can be submitted only in BG and for an inactive static partition (Fn) only.

**PARSTD=DELETE**

This option must be followed by the **filename(s)** of the DLBL or TLBL statement(s) to be deleted from the partition's standard label group. The last (or only) filename must be followed by /\*. After the /\* the option PARSTD=ADD becomes effective. PARSTD (or STDLABEL)=DELETE must be specified as the **last** option of the OPTION statement. PARSTD=DELETE is rejected if the label area does not reside on virtual disk (see USAGE=DLA operand of the VDISK statement.). DELETE may be abbreviated by DEL.

**PARSTD=(Fn,DELETE)**

This option has the same function as PARSTD=DELETE, except that it can be submitted only in BG and for an inactive static partition (Fn) only. DELETE may be abbreviated by DEL.

**PARSTD=Fn**

All label information submitted after this option will be stored into the specified partition's permanent label group. The option can be submitted only in BG, and the partition specified by Fn must be inactive.

OPTION PARSTD=Fn immediately clears the specified partition's permanent label group.

**RLD**

The assembler writes the relocation list dictionary on SYSLST. This option is suppressed if NOLIST is specified.

**NORLD**

Suppresses the RLD option.

**SADUMP=n | ([n],m)**

This option indicates the order or priority in which the partition and/or any owned data space should be dumped in a stand-alone dump. SADUMP=n controls the priority of the partition in the dump; SADUMP=(*[n]*,*m*) controls the priority (*n*) of the partition, if specified, and the priority (*m*) of any owned *data space*. Both *n* and *m* can be either 0 or 1 to 9:

- 0 Indicates that this partition or data space should not be dumped when a stand-alone dump is taken. This is also the default.

## OPTION

- 1 - 9 Indicates the priority of the partition or data space for inclusion in a stand-alone dump.

When a stand-alone dump is taken, the partition or data space with the highest priority (starting from 9) is dumped first, then the one with the next lower priority, until all partitions or data spaces for which SADUMP=0 has not been specified have been dumped (provided enough space is available on the dump device).

Example:

```
F1 ... SADUMP=(5,3)
F2 ... SADUMP=4
F3 ... SADUMP=(,9)
```

Dumps: F3-owned data space(s), F1 partition, F2 partition, F1-owned data space(s).

### SCANCEL

This option causes an operator cancel condition to be simulated whenever a job control error cancels a program without waiting for operator intervention, for example, if OPTION JCANCEL or ACANCEL and NOLOG is in effect. This allows you to specify an ON \$CANCEL command so that the job ends at the specified label. This helps to avoid that the job ending looks like a normal end-of-job.

### NOSCANCEL

Suppresses the SCANCEL option.

### SLISKIP

IF Job Control is skipping statements (due to a GOTO or IF THEN statement) and if a \* \$\$ SLI JECL statement is contained in the area to be skipped, then the \* \$\$ SLI statement is ignored. Especially if job control processes a GOTO \$EOJ (for example in case of program abend or job cancelation) this option can speed up job termination, because all POWER JECL statement are ignored.

### NOSLISKIP

IF Job Control is skipping statements (due to a GOTO or IF THEN statement) and if a \* \$\$ SLI JECL statement is contained in the area to be skipped, then the \* \$\$ SLI statement becomes effective while all other JECL statements are ignored. This is the system default (see *VSE/POWER Administration and Operation*, chapter "Interaction with z/VSE Conditional Job Control Language"). If job control processes a GOTO label then the corresponding label can be found in a library member included via \* \$\$ SLI.

### STDLABEL

This option overwrites the system standard label group. All DLBL and TLBL statements submitted after this point are written into the system standard label group to be available to all subsequent jobs in **all** partitions until another STDLABEL option without operand or with =DELETE is submitted. STDLABEL is only accepted in the BG partition.

OPTION STDLABEL remains in effect until one of the following occurs:

1. End of job or job step.
2. OPTION USRLABEL or PARSTD or CLASSTD is specified.

OPTION STDLABEL immediately clears the system standard label group (see Note 2 on page 174).

**Note:** If OPTION STDLABEL is submitted while other partitions are executing, an attempt to open a file using a standard label will result in an open failure.

#### **STDLABEL=ADD**

All label information submitted after this option will be stored into the system standard label group without overwriting the information that already exists in that label group. The label information is accessible by **all** partitions, but can only be submitted in the BG partition.

#### **STDLABEL=DELETE**

This option must be followed by the filename(s) of the DLBL or TLBL statement(s) to be deleted from the system standard label group. The last (or only) filename must be followed by `/*`. After the `/*` the option STDLABEL=ADD becomes effective.

STDLABEL=DELETE can be submitted only from BG. STDLABEL (or PARSTD) with =DELETE must be specified as the **last** option of the OPTION statement. DELETE may be abbreviated by DEL.

#### **SUBLIB=DF**

Directs the assembler and ESERV program to retrieve non-edited macros and copy-books from sublibrary members of type D instead of from sublibrary members of type A, and to retrieve edited macros from sublibrary members of type F instead of from sublibrary members of type E. IBM uses the sublibrary members of types D and F to distribute macros and copy source code for programs that are to be executed in a teleprocessing network control unit. The option remains in force until end-of-job or a `// OPTION SUBLIB=AE` statement.

#### **SUBLIB=AE**

Redirects the assembler and the ESERV program to retrieve non-edited macros and copy books from sublibrary members of type A and to retrieve edited macros from sublibrary members of type E.

#### **SYM**

The COBOL compiler produces a DATA DIVISION map on SYSLST; the PL/I compiler produces the symbol table on SYSLST.

#### **NOSYM**

Suppresses the SYM option.

#### **SYSDUMP**

Indicates that dumps are to be written to the dump sublibrary which is active for the partition. If no dump sublibrary has been defined for the partition (by a LIBDEF DUMP command), the option SYSDUMP is ignored, and the NOSYSDUMP option comes into effect. The old form of this option (SYSDMP) is accepted for compatibility reasons.

Dumps for dynamic partitions are written into the dump sublibrary SYSDUMP.DYN which is created at initial installation of the system.

#### **NOSYSDUMP**

Indicates that dumps are to be written on SYSLST. This is the default. The old form of this option (NOSYSDMP) is accepted for compatibility reasons.

#### **SYSDUMPC**

Indicates that the dump is ignored if the following two conditions are met:

- the dumps are to be written to the dump sublibrary and not to SYSLST (SYSDUMP was specified)
- the dump sublibrary is full, in error, or not defined.

## OPTION

To avoid the dump being written to SYSLST, you must specify both **SYSDUMP** and **SYSDUMPC**. Note that if you specified **NOSYSDUMP**, **SYSDUMPC** has no effect.

### **SYSPARM='string'**

Specifies a value for the assembler system variable symbol &SYSPARM. &SYSPARM gets the value of the string, which is enclosed by quotes. The string can contain 0-8 EBCDIC characters. One internal quote must be represented by two quotes. (Job control removes one of them when setting the value.) The surrounding quotes are not included and the length of &SYSPARM is determined by the resulting string.

### **TERM**

Error messages are written on SYSLOG (only applies to compilers that support this function).

### **NOTERM**

Suppresses the TERM option.

### **USRLABEL**

This option overwrites the partition's temporary label group. All DLBL and TLBL statements submitted after this point are written into the partition's temporary label group.

Specify only one of the following options XREF, SXREF and NOXREF in one OPTION statement.

### **XREF**

The assembler writes the symbol cross-reference list on SYSLST.

### **SXREF**

The assembler writes the symbol cross-reference list on SYSLST; printing of all unreferenced labels is suppressed.

### **NOXREF**

Suppresses the XREF or SXREF option.

### **48C**

Specifies the 48-character set on SYSIPT (for PL/I).

### **60C**

Specifies the 60-character set on SYSIPT (for PL/I).

### **Notes:**

1. Any assignment for SYSLNK after the occurrence of the OPTION statement cancels the LINK and CATAL options.
2. If SYSLST is assigned to a 3211 printer, the indexing feature of the device must be used with care. Shifting the print line to the left or too far to the right causes characters to be left out from every printed line of the dump.
3. If SYSLST is assigned to a 3800 Printing Subsystem, DUMP sets the 3800 to its system default for the character arrangement table, and restores the original status at the end of the dump.
4. The CLASSTD option can be used in the background only. The specified class must be disabled and no job must be active in a dynamic partition belonging to this class.
5. During program execution in a dynamic partition, the data management routines search the label information area in the following sequence:
  - a. Partitions's temporary label group
  - b. Class standard label group



c. System standard label group

If, however, the PARSTD option is used in a dynamic partition, the search sequence is the same as in static partitions:

- a. Partitions's temporary label group
- b. Partitions's permanent label group
- c. System standard label group

6. For compatibility reasons, the STDLABEL, PARSTD and CLASSTD options will **not** check for duplicate filenames if specified without the ADD operand, that is you may enter a statement like // DLBL TEST multiple times, causing multiple records for filename TEST to be written into the label group. If, however, the STDLABEL, PARSTD or CLASSTD options are specified **with** the ADD operand, then the system will check for duplicate filenames and display message 1L30D LABEL WITH SAME FILENAME IN SUBAREA if appropriate. Therefore, if you need to replace the label information for a filename you first need to DELETE the label in the corresponding label group and then ADD the new label information for filename.

## PAUSE

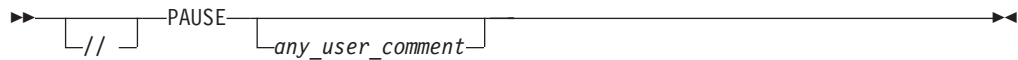
### PAUSE (Suspend Processing)

The **PAUSE statement** causes a pause immediately after processing this statement.

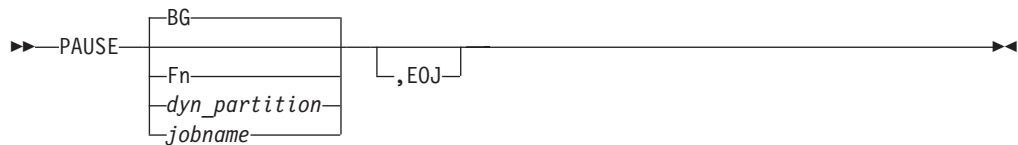
The **PAUSE command** causes a pause at the end of the current job step.

The PAUSE statement or command always appears on SYSLOG. If SYSLOG is assigned to a line printer, the PAUSE statement or command is ignored. At the time SYSLOG is unlocked for input, the operator can cause processing to be continued by pressing END/ENTER.

#### JCC, JCS Format



#### AR Format



#### BG | Fn | dyn\_partition

Indicates the static or dynamic partition in which processing is to be interrupted. If the operand is omitted, the BG partition is assumed.

#### jobname

Indicates the job name of the job to be interrupted. *jobname* can be up to 8 characters and must be unique.

#### EOJ

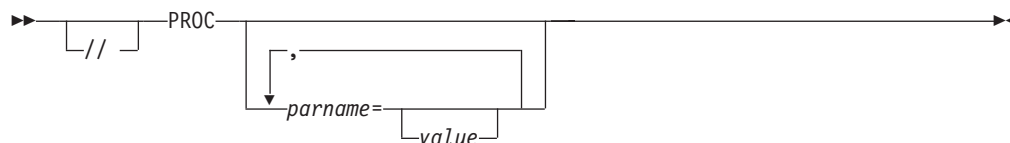
Indicates that the interruption will occur at the end of the current **job**. In this case EOJ must be preceded by a partition identifier. If the EOJ operand is omitted, the interruption will occur at the end of the current **job step**.

## PROC (Procedure)

The PROC command or statement, when used, is the first line of a cataloged procedure. It is required only when the procedure contains symbolic parameters to which you want to assign initial values. If you omit the PROC command or statement from a procedure, the system assumes a PROC statement without operands, that is, any symbolic parameters in the procedure are without a default value, and must be defined either in an EXEC PROC statement or in a SETPARM statement.

The operands of a PROC command or statement must not contain symbolic parameters. Continuation lines are accepted for the PROC statement.

### JCC, JCS Format



#### parname

The name of the symbolic parameter (without a leading &) to which you want to assign the specified value.

#### value

The value you want to assign to the specified symbolic parameter.

For rules governing the format of symbolic parameters and their values, see “Symbolic Parameters” on page 51.

**Note:** The PROC statement is accepted only within a job.

## PRTY (Query and Set Partition Priorities)

The AR PRTY command allows the operator to:

- Display the priority sequence of the static partitions or dynamic partition classes in the system;
- Change that sequence for one, some or all partitions or classes.

In both cases also the current status (if active) of the TP Balancing (TPBAL) function is displayed.

The JCC PRTY command can be used only in the BG during ASI (Automated System Initialization) to modify the priority sequence of the partitions in the system.

Continuation lines are accepted for the PRTY command.

### AR Format

»» PRTY \_\_\_\_\_

### AR, JCC Format

»» PRTY 

BG
F <sub>n</sub>
class

 \_\_\_\_\_

### AR, JCC Format

»» PRTY 

BG
F <sub>n</sub>
class

 , 

,BELOW
,EQUAL

 , 

BG
F <sub>n</sub>
class

 \_\_\_\_\_

### AR, JCC Format

»» PRTY 

BG
F <sub>n</sub>
class

 = 

,BELOW
,EQUAL

 , 

BG
F <sub>n</sub>
class

 \_\_\_\_\_

where

BG, F<sub>n</sub> is one of the 12 static partitions BG, F1...FB  
 'class' is one of the 23 classes of dynamic partitions.

The AR PRTY command **without** an operand displays, on SYSLOG, the current processor dispatching priorities of all partitions and classes (if available). The output consists of a list in which the entries are separated by a comma or, if the entries belong to a balanced group, by an equal sign (=). The first partition in the list has the lowest priority, the last one the highest. Partitions which are members of a balanced group have the same priority.

The following example shows a currently active PRTY string:

```
PRTY FB,Q=FA=F9=F8=F7,F6,P,F5,F4,BG,N,F3,F2,F1
```

where Q, P, and N are classes of dynamic partitions

In this example, FB has the lowest priority, F1 the highest. The dynamic partition class Q and the partitions FA, F9, F8, and F7 belong to a balanced group with a single (the second-lowest) priority.

When balancing a dynamic class, the entire class priority is addressed, so that when n partitions are active in the class, each of them will get 1/n of CPU time. In the above example four partitions are running in class Q.

For the z/VSE **Turbo Dispatcher**, the distributed time is as follows: 1 unit each for FA, F9, F8, and F7; 1 each for Q1, Q2, Q3, and Q4.

The PRTY command with **one** operand only indicates that the specified partition or class receives the highest priority. For example,

```
PRTY F3
```

The PRTY command with **two or more** operands can be specified either with a comma or with an equal sign as separator, or in a mixed form to provide for priority setting and partition balancing together, as indicated below.

**BG | Fn | class,BG | Fn | class,BG | Fn | class...**

Indicates the desired sequence of processing priority **within the specified string**. The first partition/class you specify receives the lowest priority, the last one the highest priority. For example,

```
PRTY BG,F4,F2,F1
```

Of these four partitions, the background partition receives the lowest priority and F1 the highest.

Note, however, that missing partitions or classes will always receive the lowest priority and the same sequence order as in a previously valid priority string. Thus in the above example, the sequence **BG,F4,F2,F1** will be put at the top of the priority list, with BG having the lowest priority only within the specified list.

**BG | Fn | class=BG | Fn | class=BG | Fn | class...**

Specifies that partition balancing is to be used for the partitions or classes which you list with a separating equal sign (=). Partitions so specified are treated as an entity within which the supervisor checks processor usage at regular intervals and reassigns priorities such that the partition with the highest processor usage is given lowest priority.

**Mixed format:**

The command

```
PRTY BG,F2=F3=F4,F5,F6,F1
```

specifies highest priority for partition F1, lowest priority for partition BG, and partition balancing for partitions F2 to F4.

**BELOW**

Specifies that the partitions/classes specified before the keyword BELOW get the next lower priority to the partition/class following BELOW.

## PRTY

### EQUAL

Specifies that the partitions/classes specified before the keyword EQUAL are combined to a balanced group with the partition/class specified after EQUAL.

For example, if the actual PRTY sequence is

```
FB,FA,N,F9,F8,F7,Q,F6,F5,F4,BG,P,F3,F2,F1
```

the command

```
PRTY BG=N=FA,FB,BELOW,F6
```

results in the new PRTY sequence:

```
F9,F8,F7,Q,BG=N=FA,FB,F6,F5,F4,P,F3,F2,F1
```

The next command

```
PRTY F3,F4=F5,EQUAL,N
```

results in the new PRTY sequence:

```
F9,F8,F7,Q,F3,F4=F5=BG=N=FA,FB,F6,P,F2,F1
```

### Notes:

1. If VTAM is used, the partition in which it is running should not be specified for partition balancing.
2. If teleprocessing balancing is active, the partition(s) that can be subject to deactivation will be displayed on SYSLOG, regardless of whether operands are specified with the PRTY command or not. See also the TPBAL command.
3. A VSE/POWER partition (normally F1) should have a higher priority than the POWER-controlled partitions. If you prefer to give VSE/POWER a lower priority, you can do this both for static and dynamic partitions. For static partitions the NPC (no priority check) operand of the PSTART command must be used in this case.
4. You can specify only one group of partitions for partition balancing.
5. Only such dynamic classes can be specified which are contained in the active class table. Therefore, the VSE/POWER PLOAD command for a dynamic class table has to be processed before the PRTY command can set the priorities of dynamic classes.

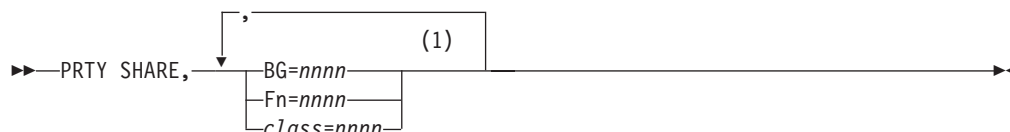
The priority setting may be changed via the PLOAD command. New classes get lowest priority. Classes that do not exist in the newly loaded dynamic class table are removed from priority handling.

## PRTY SHARE Command

The PRTY SHARE command is used to allocate a relative share of CPU time to partitions belonging to a balanced group. The relative share of CPU time for a partition is reflected by a numeric value. Such a value can be defined for a static partition or for a dynamic class.

The PRTY SHARE command is available as AR (attention routine) command and as job control command in the startup procedure for the BG partition. The command is ignored if the Turbo Dispatcher is not active.

### AR, JCC Format



#### Notes:

- 1 Each partition or dynamic class can be specified only once.

#### SHARE

Indicates that this PRTY command applies to static partitions and dynamic classes of a **balanced group**. The values for the relative share of CPU time to be allocated are to be changed or newly defined.

#### BG, Fn, class

Defines either one of the static partitions BG, or F1 through FB, or a dynamic class. The static partitions or the dynamic classes specified must be included in the priority sequence (the sequence you get when you enter the PRTY command without operands).

Although you can specify a share value for each static partition or each dynamic class shown in the priority sequence, a share value becomes effective only if the static partition or dynamic class belongs to a balanced group.

#### nnnn

Defines the numeric value which determines the relative share of CPU time allocated to a static partition or a dynamic class. For dynamic partitions this means that each dynamic partition belonging to the same class gets the same value allocated. Operand *nnnn* can range from 0 to 9999. The default is 100.

If a balanced group includes two active partitions where partition A has a relative share of 100 and partition B a relative share of 200, then partition B gets twice as much CPU time allocated than partition A. The same effect can be achieved, for example, by specifying 1 for partition A, and 2 for partition B.

A share value of 0 implies that this partition or class will no longer participate in partition balancing and will be moved to the lowest priority within the balanced group. A member of a balanced group with a share value of 0 *will not* receive any time slice unless all other members with a share value greater than 0 are in a wait state. However, a member of a balanced group with a share value of 1 *will* receive a time slice no matter what share values have been specified for the other members of the group.

## PRTYIO (Query and Set Partition Priorities for I/O Requests)

The PRTYIO command allows the operator to set priorities for the handling of I/O requests for your partitions. It may be used, for example, to give your CICS partition(s) I/O priority over batch partitions. If batch and CICS access the same logical devices, this may improve terminal response time while the impact on batch throughput is minimal.

### AR Format

▶▶ PRTYIO ◀◀

### AR Format

▶▶ PRTYIO ◀◀  
           ┌───┐  
           | , |  
           └───┘  
           ┌───┐  
           | BG |  
           └───┘  
           ┌───┐  
           | Fn |  
           └───┘

### AR Format

▶▶ PRTYIO ◀◀  
           ┌───┐  
           | = |  
           └───┘  
           ┌───┐  
           | BG |  
           └───┘  
           ┌───┐  
           | Fn |  
           └───┘

### AR Format

▶▶ PRTYIO DYNC ◀◀

### AR Format

▶▶ PRTYIO OFF ◀◀

The command can be given in any combination of the second, third, and fourth formats.

The command **without** an operand displays, on SYSLOG, the current priorities for I/O requests for your partitions. If no priorities have been set, the command causes the message **PRTYIO NOT SET** to be displayed. The default value for **PRTYIO NOT SET** is first-come, first-served, independent of the partition priorities. System tasks have highest priority.

### BG | Fn

The command with one or more **partition** specifications sets priorities for the handling of I/O requests from static partitions. I/O requests from the static partition specified first in the list are handled with highest priority; the requests from the partition specified next are handled with next lower priority, and so on. Note that this priority sequencing is in direct contrast to that of the PRTY command, where the first partition has the lowest priority and the last one the highest.

The effect of two or more PRTYIO commands during a system run is cumulative. This is best explained by examples:



1. No PRTYIO command is specified: the I/O requests for a device are handled first-in first-out.

2. The command

```
PRTYIO F1,F4
```

causes I/O requests for a device from partition F1 to be handled with highest priority and from partition F4 with second-highest priority. I/O requests for the device from any other partition are handled first-in first-out.

3. The command

```
PRTYIO F3=F4
```

issued after the command in example 2 results in I/O priorities to be set for a device as listed below.

- Highest: requests from F3 and F4; these requests are handled first-in first-out.
- Next lower: requests from F1.
- Next lower (actually the lowest): requests from the remaining partitions; these requests are handled first-in first-out.

4. The command

```
PRTYIO F2,F3
```

issued after the commands in examples 2 and 3 causes the I/O priorities to be set as shown:

- Highest: requests from F2.
- Next lower: requests from F3.
- Next lower: requests from F4.
- Next lower: requests from F1.
- Next lower (actually the lowest): requests from the remaining partitions; these requests are handled first-in first-out.

**DYNC**

The PRTYIO command with the **DYNC** operand sets the priority of all **dynamic** partitions. This priority applies to all dynamic partitions in the system. For example, the command

```
PRTYIO F1,DYNC
```

causes I/O requests for a device from the static partition F1 to be handled with highest priority and from all the dynamic partitions with second-highest priority.

**OFF**

I/O priority specifications are reset either by the command

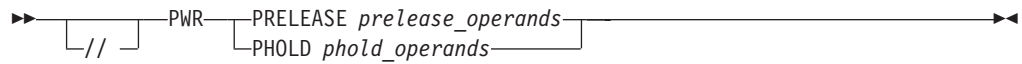
```
PRTYIO OFF
```

or during the next startup of your system.

## PWR (Pass POWER Command)

The PWR job control statement makes it possible to pass the commands PRELEASE and PHOLD to POWER at any point in the job stream. The operand of the PWR statement is taken as a POWER command, and its syntax is checked by the POWER routine.

### JCC, JCS Format



### PWR

Specifies that the rest of the statement is a POWER command.

### PRELEASE | PHOLD

Are the POWER commands which will be accepted. For their syntax requirements, see the applicable VSE/POWER publication.

### Notes:

1. The second operand of the POWER command must not be a single character (CLASS) or ALL.
2. Symbolic parameters cannot be used in this statement.

## PWROFF (Power Off CPU)

The PWROFF command allows the system operator to power off the CPU, provided the CPU (for example, an IBM 4300 or 9370) has the Programmed Power Off feature. If the CPU does not have this feature, the command is invalid.

### AR Format

▶▶—PWROFF—————▶▶

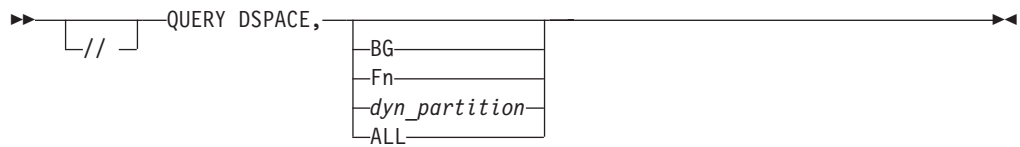
The PWROFF command has no operand.

## QUERY (Query Data Spaces and Standard Options)

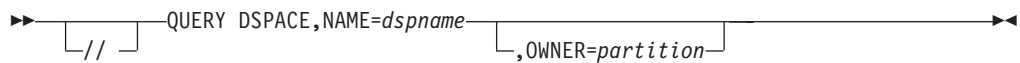
The QUERY command/statement can be used to display information on the following:

- Data spaces (DSPACE)
- Standard options (OPTIONS, STDOPT)
- Symbolic parameters (SETPARM)
- The SCSI configuration (see “QUERY SCSI” on page 191 for details)
- The z/VSE multiprocessor environment (see “QUERY TD” on page 192 for details)

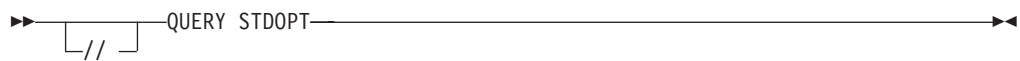
### AR, JCC, JCS Format



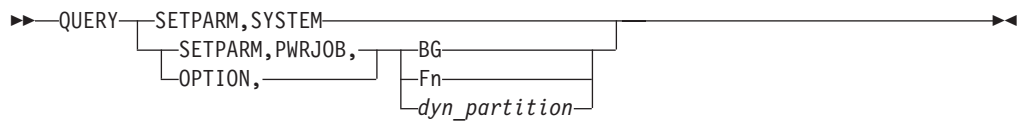
### AR, JCC, JCS Format



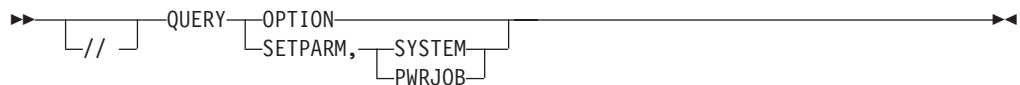
### AR, JCC, JCS Format



### AR Format



### JCC, JCS Format



### DSPACE (without operand)

Displays summary information on data spaces.

### BG | Fn | dyn\_partition

Displays detailed information on data spaces for the specified (static or dynamic) partition.

### ALL

Displays detailed information on all data spaces in the system.

### NAME=dspname

Displays detailed information on the data space named **dspname**.

**OWNER=partition**

Displays detailed information on the data space **dspname** (specified in the NAME operand) which is owned by the specified partition.

**STDOPT**

Causes the current setting of all standard options to be displayed on the console.

**OPTION**

Causes the current setting of all temporary options to be displayed on the console.

**SETPARM**

Causes currently defined symbolic parameters to be displayed on the console.

**SYSTEM**

Specifies that symbolic parameters at system level are displayed on the console.

**PWRJOB**

Specifies that symbolic parameters at POWER job level are displayed on the console.

**BG | Fn | dyn\_partition**

Denotes a static or dynamic partition, for which detailed information on one of the following entities is displayed:

- data spaces
- temporary job control options
- symbolic parameters at POWER job level

**Output of QUERY DSPACE**

---

```

0 QUERY DSPACE
BG 0000                DSIZE    MAX  PARTMAX    COMMAX  VDISK   DFSIZE
BG 0000  DEFINED:  xxxxxxxxxK  nnnnn    mmm      kkk    nnn  xxxxxxxK
BG 0000  ACTUAL:  xxxxxxxxxK  nnnnn    mmm      kkk    nnn
BG 0000
BG 0000  AREA DSPTS  AREA DSPTS  AREA DSPTS  AREA DSPTS  AREA DSPTS  AREA DSPTS
BG 0000  pp   nnn   pp   nnn   pp   nnn   pp   nnn   pp   nnn   pp   nnn
BG 0000  pp   nnn   pp   nnn   pp   nnn   pp   nnn   pp   nnn   pp   nnn
BG 0000  pp   nnn   pp   nnn   pp   nnn   pp   nnn   pp   nnn   pp   nnn
BG 0000  pp   nnn   pp   nnn   pp   nnn   pp   nnn   pp   nnn   pp   nnn
BG 0000

```

---

where

**DSIZE, MAX, PARTMAX, COMMAX, DFSIZE**

In the line DEFINED: show the values as defined with the SYSDEF command. This is illustrated in Figure 7 on page 188. In the line ACTUAL: show the current values (not applicable for DFSIZE).

**Note:** For DSIZE, MAX, and PARTMAX the actual values may be higher than the limits defined in the SYSDEF command.

The actual value for DSIZE is the amount of virtual storage that has been taken from VSIZE for the currently allocated data spaces.

**VDISK**

in the line DEFINED: shows the number of virtual disks added at IPL time

## QUERY

(ADD cuu,FBV) and not overwritten by IPL device sensing; in the line ACTUAL: shows the number of virtual disks allocated via the VDISK job control command.

**Note:** For each defined virtual disk, an entry is reserved in each primary address space access list (PASN-AL).

### AREA

'ppp' may be SYS or a partition-id. The sequence of displayed areas is: SYS, BG, F1, ..., FB, dynamic partitions of class C, ..., dynamic partitions of class Z. Only those areas are shown which own at least one data space.

### DSPS

Number of data spaces owned by AREA.

```
QUERY DSPACE
AR 0015          DSIZE  MAX  PARTMAX  COMMAX  VDISK  DFSIZE
AR 0015 DEFINED: 20480K 256   16      5        3      960K
AR 0015 ACTUAL:  9856K   6     3       1        2
AR 0015
AR 0015 AREA DSPS  AREA DSPS  AREA DSPS  AREA DSPS  AREA DSPS
AR 0015 BG      2   FB      1   F3      3
```

Figure 7. Example of QUERY DSPACE Output

### Output of QUERY DSPACE,F3

With this command format information is displayed for data spaces which are created and/or accessed by the named partition. 'Accessed' means: the data space has an entry in a DU-AL of one of the partition's (sub)tasks or in the PASN-AL of the address space where the partition is allocated.

```
QUERY DSPACE,F3
AR 0015 AREA DSPNAME  SIZE  MAXSIZE  SCOPE  OWNER  DU-AL  PASN-AL
AR 0015 F2  ISTD7B66  1024K  2048K  ALL    F3      X
AR 0015
AR 0015 F3  SYSIVFDF  1440K  1440K  COMMON  BG      X
AR 0015 F3  SYSIV9F0  3360K  3360K  COMMON  BG      X
AR 0015 F3  BSMSPACE  960K   960K   COMMON  FB      X
AR 0015 F3  ISTAA114  1024K  2048K  ALL    F3      X
AR 0015 F3  ISTD7B66  1024K  2048K  ALL    F3      X
AR 0015 F3  IST2A6E0  2048K  4096K  ALL    F3      X
AR 0015
AR 0015 F7  IST2A6E0  2048K  4096K  ALL    F3      X
```

Figure 8. Example of QUERY DSPACE,F3 Output

where

### AREA

Partition accessing the data space. Different partitions are separated from each other by a blank line.

### DSPNAME

Name of the data space (as defined by the DSPSERV CREATE macro).

### SIZE

Currently allocated data space size. This value is rounded up to the next multiple of 32K and is taken from VSIZE.

### MAXSIZE

Maximum size of the data space (as defined by the DSPSERV CREATE macro).

**SCOPE**

Scope of the data space. May be SINGLE, ALL, or COMMON (as defined by the DSPSERV CREATE macro).

**OWNER**

Owner of the data space. This may be the partition itself, or the partition-id of another partition, or SYS (any system task).

**DU-AL**

Shows 'X' if the data space has an entry in a DU-AL of at least one task of the accessing partition.

**PASN-AL**

Shows 'X' if the data space has an entry in the PASN-AL of the accessing partition.

If no data space exists which is created by or accessed by a partition, a message is displayed.

**Output of QUERY DSPSPACE,ALL**

Displays the same information as QUERY DSPSPACE,BG|FN|dyn\_partition, except that QUERY DSPSPACE,ALL includes information for **all** partitions which have created data spaces and/or have access to data spaces. For data spaces with SCOPE=COMMON, the field AREA is left blank and only the owning partition is shown. The different partitions are separated from each other by a blank line.

0 QUERY DSPSPACE,ALL									
BG	0000	AREA	DSPNAME	SIZE	MAXSIZE	SCOPE	OWNER	DU-AL	PASN-AL
BG	0000		SYSIVFDF	1440K	1440K	COMMON	BG		X
BG	0000		SYSIV9F0	3360K	3360K	COMMON	BG		X
BG	0000		BSMSPACE	960K	960K	COMMON	FB		X
BG 0000									
BG	0000	F2	ISTD7B66	1024K	2048K	ALL	F3		X
BG 0000									
BG	0000	F3	ISTAA114	1024K	2048K	ALL	F3		X
BG	0000	F3	ISTD7B66	1024K	2048K	ALL	F3		X
BG	0000	F3	IST2A6E0	2048K	4096K	ALL	F3		X
BG 0000									
BG	0000	F7	IST2A6E0	2048K	4096K	ALL	F3		X

Figure 9. Example of QUERY DSPSPACE,ALL Output

**Output of QUERY DSPSPACE,NAME=dspname[,OWNER=partition]**

Displays information for the named data spaces.

The output is similar to the output of QUERY DSPSPACE,ALL.

QUERY DSPSPACE,NAME=dspname shows all lines with DSPNAME=dspname (sorted by the AREA column).

QUERY DSPSPACE,NAME=dspname,OWNER=partition shows all lines with DSPNAME=dspname and OWNER=partition.

**Output of QUERY SETPARM,PWRJOB**

The symbolic parameters are displayed according to the sequence of the corresponding SETPARM commands (re-)assigning their values. Both R1PARAM1 and R1PARAM4 have the null string as value, therefore no value information is displayed on the right hand side of the equal sign.

## QUERY

```
QUERY SETPARM PWRJOB,R1  
AR 0015 R1P2    = SECOND PARAMETER  
AR 0015 R1P3    = THIRD PARAMETER  
AR 0015 R1P4    =  
AR 0015 R1P5    =
```

*Figure 10. Example of QUERY SETPARM,PWRJOB Output*



## QUERY SCSI

### AR, JCC, JCS Format



Use the QUERY SCSI command to query all SCSI devices in the system and their characteristics. To obtain the configuration of a single SCSI device in the system, use QUERY SCSI ,cuu. Figure 11 shows an output example.

#### cuu

Indicates the FBA device for which SCSI/FCP related information is to be displayed. If this operand is omitted, the information is displayed for all FBA devices defined with a preceding SYSDEF SCSI (or IPL DEF SCSI) command. The output is sorted by ascending FBA device numbers (FBA-CUU). In case of a multi-path definition (that is the FBA-CUU is connected to the LUN via different FCP-CUU's), the secondary path(s) is flagged with MP right behind the FBA-CUU number.

```

QUERY SCSI,500
AR 0015 FBA-CUU FCP-CUU WORLDWIDE PORTNAME LOGICAL UNIT NUMBER
AR 0015 500 C00 5005076300CB93CB 5178000000000000
AR 0015 500MP C01 5005076300CB93CB 5178000000000000
    
```

Figure 11. Output Example of QUERY SCSI Command

The information displayed by QUERY SCSI has the following meaning:

#### FBA-CUU

The SCSI device, defined as FBA device via the ADD command during IPL

#### FCP-CUU

The FCP adapter by which the SCSI device is attached, defined via the ADD command during IPL.

#### WORLDWIDE PORTNAME

The 64 bit world wide port name of the SCSI controller configured to access the LUN. It is specified in 16 hexadecimal digits. Valid specifications are 0 to 9 and A to F.

#### LOGICAL UNIT NUMBER

The 64 bit logical unit number identifying the particular SCSI device as configured in the SCSI controller. It is specified in 1 to 16 hexadecimal digits. Valid specifications are 0 to 9 and A to F. If less digits than 16 are specified, trailing zeros will be presumed. For example, LUN 216B0000 00000000 can be specified as LUN=216B.

## QUERY TD

See the *VSE/ESA Turbo Dispatcher Guide and Reference* for detailed information.

### AR, JCC, JCS Format



To query the status of a z/VSE multiprocessor environment, you can use the QUERY TD

command. Figure 12 shows an output example.

```

QUERY TD
AR 0015 CPU      STATUS      SPIN_TIME      NP_TIME  TOTAL_TIME NP/TOT
AR 0015 00      ACTIVE        5656           40856    147691  0.276
AR 0015 01      ACTIVE        4675           39930    148688  0.268
AR 0015 02      INACTIVE
AR 0015 03      ACTIVE        4619           39940    148734  0.268
AR 0015 -----
AR 0015 TOTAL          14950          120726    445113  0.271
AR 0015
AR 0015              NP/TOT: 0.271      SPIN/(SPIN+TOT): 0.032
AR 0015  OVERALL UTILIZATION: 294%      NP UTILIZATION: 77%
AR 0015
AR 0015  ELAPSED TIME SINCE LAST RESET:      156094
AR 0015 1140I  READY
    
```

Figure 12. Output Example of QUERY TD Command

The information displayed by QUERY TD has the following meaning:

**CPU** Shows the CPU address; also referred to as CPU number. The CPU address is assigned during system installation. The first CPU displayed is the CPU from which IPL was performed.

**STATUS** Displays the current state of each CPU. This status field contains either ACTIVE, INACTIVE or QUIESCED. For information on how to change the CPU state refer to the *VSE/ESA Turbo Dispatcher Guide and Reference*.

**SPIN\_TIME** Shows the time in milliseconds during which the CPU was within an instruction loop waiting for a resource occupied by another task.

**NP\_TIME** Shows the time in milliseconds during which the CPU processed non-parallel work units. Only one non-parallel work unit can be processed at a time. As long as a CPU processes a non-parallel work unit, the other CPUs can process parallel work units only.

The NP\_TIME value is included in the TOTAL\_TIME value.

**TOTAL\_TIME** Shows the time in milliseconds during which the CPU processed either parallel or non-parallel work units. This means that the TOTAL\_TIME value includes the NP\_TIME value. Note that the TOTAL\_TIME does not include the SPIN\_TIME.

**NP/TOT**

Shows the ratio of non-parallel time to total time (the quotient out of NP\_TIME and TOTAL\_TIME). The smaller the ratio, the higher is the potential for exploiting more CPUs.

**Note:** This value represents the **non-parallel share** (NPS) of a workload. It can be used for a rough estimate of the number of CPUs required to process efficiently the current workload mix. If, as in the example, the NP/TOT ratio is approximately 0.3 (0.271), then the related workload or workload mix can fully exploit 3 CPUs.

If the TOTAL\_TIME value is zero or exceeds the maximum (see note below), then **\*.\*\*\*** is displayed as the NP/TOT ratio.

**TOTAL**

Shows the total sum or average value for each column.

Further values and information displayed are:

- **NP/TOT**

This is a repetition of the average value of the NP/TOT column. See previous description for the meaning of this value.

- **SPIN/(SPIN+TOT)**

This value is calculated as follows:  $SPIN\_TIME / (SPIN\_TIME + TOTAL\_TIME)$

The higher this value, the more time the CPU was waiting for resources occupied by other tasks.

- **OVERALL UTILIZATION**

This value is calculated as follows:

$100 \times (TOTAL\_TIME + SPIN\_TIME) / ELAPSED\_TIME$

This utilization is the sum of all utilizations of all individual processors and can thus add up to  $n \times 100\%$ .

- **NP UTILIZATION**

This value is calculated as follows:

$100 \times NONPARALLEL\_TIME / ELAPSED\_TIME$

The resulting value reflects the utilization of the NP (non-parallel) status and can thus reach at most 100%. It is a good indicator of the remaining potential for exploiting more processors.

- **ELAPSED TIME SINCE LAST RESET**

Shows in milliseconds the time passed since the last reset of CPU related information. Such a reset occurs whenever a SYSDEF TD command or statement is being processed.

**Note:** In case of numerical overflow, the number fields are padded with \*. For example, if ELAPSED\_TIME gets higher than 2147483647 (corresponding to a time period of approximately 25 days), the OVERALL UTILIZATION and NP UTILIZATION will be displayed as **\*\*\*%**.

## RC (Request Communication)

The operator can use the RC command if he wants to enter an AR command when the attention routine is not available. In this case, entering the RC command forces the system to:

- Terminate processing of any previous attention routine command, and
- Accept any new attention routine command from the console.

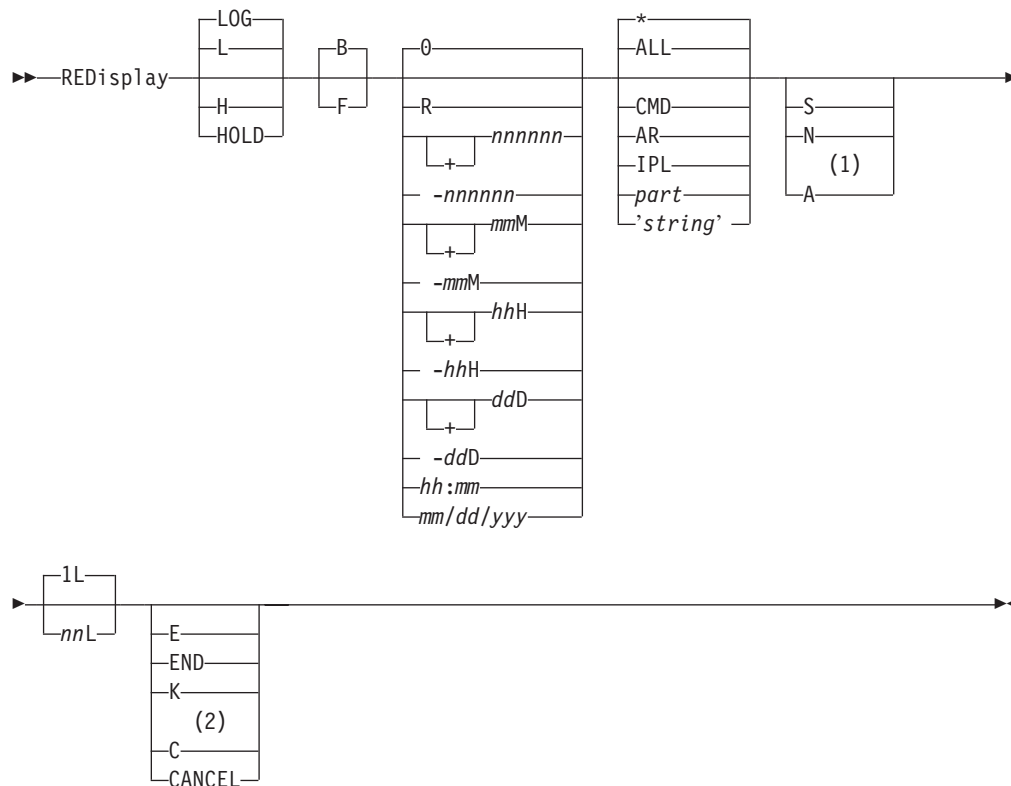
### AR Format

▶▶—RC—————▶▶

The RC command has no operand.

## REDISPLAY (Retrieve Logged Information)

The REDISPLAY command lets you retrieve logging information that had earlier been displayed on one or more consoles. This information consists of *logging items* such as messages issued by the system, or commands that you entered and the system's response to these commands.



**Notes:**

- 1 Together with H/HOLD, a subfilter cannot be specified.
- 2 Together with C/CANCEL, no other operand can be specified.

You enter the REDISPLAY command in one of two ways:

- As **system command** (AR Format)
- As **local command**

In this case, you have to prefix the percent character: %REDISPLAY. It is advisable to assign a PF key to the %REDISPLAY command. Normally PF7 is assigned to the %REDISPLAY command. When using the PF key, you may or may not supply some operand(s) in the input line.

When the %REDISPLAY command is given in console mode, the console goes into **redisplay mode**. While in redisplay mode, you may issue additional %REDISPLAY commands, preferably by using some PFkey function.

All operands are optional. **The operands can be specified in any sequence and must be separated by a comma.**

If a **default** is listed for an operand, it applies to the time when the console is not yet in redisplay mode. When the console is already in redisplay mode, the values

## REDISPLAY

of the preceding Redisplay command are taken as default. However, **startpos** (the third column in the diagram) always has a default of 0.

A change of **function** (the first column in the diagram) causes the defaults for the other operands to be chosen as if the console were not yet in redisplay mode.

The parameter values are described below. They are grouped into the following items:

- functions
- direction
- startpos
- filter
- subfilter
- lines
- action.

### **function**

specifies the scope of data to be redisplayed:

**L**

**LOG**

requests redisplay of any kind of logging items. This is the default.

**H**

**HOLD**

requests only redisplay of messages that are waiting for a reply or for an action.

### **direction**

determines whether the redisplay moves forward or backward:

**B** the direction of redisplay is backward. This is the default.

**F** the direction of redisplay to forward.

### **startpos**

specifies where the redisplay is to start:

**R** causes redisplay to be **restarted** from the point where redisplay mode was entered.

**[+]nnnnnn**

**-nnnnnn**

specifies the number of lines to be spaced forward (+) or backward (-) starting at the current position. startpos has as default a value of 0 for nnnnnn.

**[+]mmM**

**-mmM**

specifies the number of minutes that is to be added (+) to or subtracted (-) from the time of the current position.

**[+]hhH**

**-hhH**

specifies the number of hours that is to be added (+) to or subtracted (-) from the time of the current position.

**[+]ddD**

**-ddD**

specifies the number of days that is to be added (+) to or subtracted (-) from the date of the current position.

**hh:mm**

requests that the redisplay is to start at the message with the specified time (of the current day).

Leading zeros have to be specified.

**mm/dd/yyyy**

requests that the redisplay is to start at the message with the specified date (you may indicate the year by only two digits 'yy', leaving out the century).

Leading zeros have to be specified.

**filter**

specifies selection criteria:

**ALL**

\* requests that the set of logging items is not to be restricted in any way. This is the default.

**CMD**

requests redisplay of all entered commands (Attention, VSE/POWER, VM, CP, invalid commands) together with the system's responses to these commands. For example, if 'D RDR' had been entered, not only the 'D RDR' command but also the related responses are redisplayed.

**AR**

requests redisplay of Attention Routine commands together with the system's responses to these commands.

**IPL**

requests redisplay of all commands entered during IPL and their command responses. Only items up to the message

```
0I20I  IPL COMPLETE FOR...
```

are displayed.

**part**

requests redisplay of all logging items that belong to a specific partition. **part** designates a static partition (BG, Fn), a dynamic partition (U2, for example), or a class of dynamic partitions, which is indicated by an asterisk in the second position (U\*, for example).

**'string'**

requests redisplay of all messages and replies that contain the specified character-string within one line. char-string may be up to 15 characters long.

**subfilter**

allows to specify a second selection filter in addition to the one specified in **filter**:

**S** requests redisplay of all messages which were suppressed or replied to by an operator-automation product.

**N** requests redisplay of all logging items directed to or entered at an operator-automation console.

**A** requests redisplay of action messages.

The subfilter can be turned off by entering a new **filter**.

**Note:** For the H (HOLD) function, a subfilter cannot be specified.

## REDISPLAY

### **lines**

specifies the number of lines to be redisplayed.

### **nnL**

for **nn** any value value between 1 and 99 is allowed. 1 is the default.

The last message is displayed in its entirety. Therefore up to 11 lines above the specified lines value may appear.

### **action**

specifies something about ending redisplay processing:

### **E**

### **END**

requests that the redisplay mode is to be ended.

**K** This is the **Keep** option. The position on the hardcopy file where redisplaying starts will be preserved for the next redisplay request.

### **C**

### **CANCEL**

specifies that the Redisplay command **currently in process** is to be cancelled immediately. If no Redisplay command is in progress, this command has no effect. No other operands are allowed when the **CANCEL** action is requested.



## REPLID (Query Reply-IDs)

The REPLID command displays the reply-IDs (and partition indicators) of all messages for which replies are still pending.

For information on how to use this command, refer to “Handling System Messages” in the manual *z/VSE Operation*.

### AR Format

▶▶—REPLID—▶▶

The REPLID command has no operand.

## RESERV

### RESERV (Reserve Device for VSE/VSAM)

The RESERV command reserves a device for VSE/VSAM space management usage. This means that the device cannot be assigned any more in the system. Also, a DVCDN command for the device will be rejected. The reserved status can be reset only by a FREE command.

The command may be issued for all disk devices on the system.

#### AR Format

►►—RESERV *cuu*—————►

#### **cuu**

Indicates the device number of the device to be reserved.

## RESET (Reset ASSGNs and LIBDEFs to Permanent Values)

The RESET command or statement resets temporary sublibrary definitions (LIBDEFs) and I/O assignments to their permanent values in the partition in which RESET was submitted. For information on temporary and permanent assignments and sublibrary definitions, refer to the ASSGN and LIBDEF statements.

When the physical device affected by RESET is a magnetic tape drive, the current mode set in the PUB table is set to the standard mode set for the device. The standard mode set is established during IPL and may be modified by a permanent ASSGN with a mode operand.

### JCC, JCS Format



#### SYS

Resets all system logical unit assignments and library search chain definitions to their permanent values.

#### PROG

Resets all programmer logical units to their permanent assignments.

#### ALL

Resets all logical unit assignments and library chain definitions to their permanent values.

#### SYSxxx

Resets the specified logical unit to its permanent assignment. SYSIN or SYSOUT cannot be specified.

## ROD

### ROD (Record on Demand)

The ROD command records all statistical data record counters for all non-telecommunication devices on the recorder file on SYSREC. The buffer containing the last console messages is written to the hardcopy file. The command must not be issued until all jobs in the partitions have finished executing.

#### JCC Format

▶▶—ROD—————▶▶

The ROD command has no operand.

## RSTRT (Restart Checkpointed Program)

The RSTRT statement is available for checkpointed programs. A programmer can use the CHKPT macro instruction in his program to write checkpoint records. The maximum number of checkpoints that can be taken is decimal 9999. The checkpointed information includes the registers, tape-positioning information, a dump of the program, and a restart address.

The restart facility allows the operator to continue execution of an interrupted job at a point other than the beginning. To do so, submit a RSTRT command followed by the job control statements originally used for the job.

The RSTRT statement is not allowed in a dynamic partition.

### JCS Format

```

▶▶—// RSTRT SYSxxx,nnnn—┬──────────────────────────────────────────▶▶
                          │,filename│
  
```

#### SYSxxx

Logical unit name of the device on which the checkpoint file is stored. This unit must have been previously assigned.

#### nnnn

Identification of the checkpoint record to be used for restarting. It corresponds to the checkpoint identification used when the checkpoint was taken. The serial number is supplied by the checkpoint routine and printed on SYSLOG when the checkpoint was taken.

#### filename

The name of the disk checkpoint file to be used for restarting. It must be identical to the filename of the DTFPH to describe the disk checkpoint file and the fifth parameter of the CHKPT macro instruction. This operand only applies when specifying a disk as the checkpoint file.

See “CHKPT Macro” in the manual *z/VSE System Macros Reference* for further details on the CHKPT macro instruction.

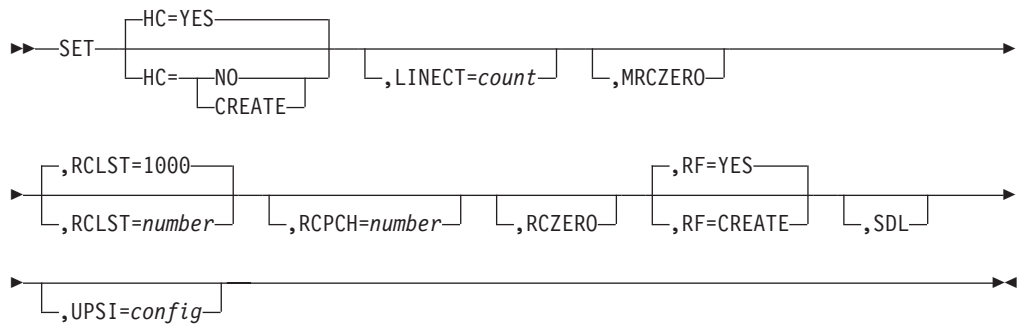
When a checkpoint is taken, the completed checkpoint is noted on SYSLOG. Restarting can be done from any checkpoint record, not just the last. The jobname specified in the JOB statement must be identical to the jobname used when the checkpoint was taken. The proper I/O device assignments must precede the RSTRT control statement.

Assignment of input/output devices to logical unit names may vary from the initial assignment. Assignments are made for restarting jobs in the same manner as assignments are made for normal jobs. Run mode (virtual or real), and storage allocations and boundaries for the partition must be the same for the restart run as for the original, checkpointed run.

## SET (Set Program Control Values)

The SET command sets controls for the execution of programs. Except for SET UPSI and SET MRCZERO/RCZERO, the SET command should precede the JOB statement of the first job for which the specified control value is to be effective.

### JCC Format



#### HC=YES | NO | CREATE

Defines the status of the hardcopy file (IJSYSCN) on SYSREC. It may be specified only after IPL and before the first JOB statement.

##### YES

Indicates that a hardcopy file exists in the system, and that it is to be opened. This is the initial system default.

##### NO

Indicates that no recording is to be performed on the hardcopy file. Can be specified only if a console printer is attached.

##### CREATE

Instructs the system to create a hardcopy file; the file is created and opened as soon as the first JOB statement is read.

The HC parameter is not allowed in a dynamic partition.

#### LINECT=count

Sets the standard number of lines to be printed on each page of SYSLST. Specify an integer between 30 and 160.

#### MRCZERO

Re-initializes the maximum return code (as the JOB or /& statement would).

#### RCLST=number

Specifies the number of records remaining to be written on a SYSLST extent on disk before a warning is issued to the operator that the extent is nearly full. Specify any decimal number from 100 through 65535. The initial system value is 1000. RCLST is ignored if SYSLST is assigned to a diskette.

**Note:** This warning is issued only between job steps. If the extent limits are exceeded before the job step ends, this job is terminated.

The RCLST parameter is not allowed in a dynamic partition.

#### RCPCH=number

Specifies the minimum number of records remaining to be written on a SYSPCH extent on disk before a warning is issued to the operator that the

extent is nearly full. It may be any decimal number from 100 through 65535. The initial system value is 1000. RCPCH is ignored if SYSPCH is assigned to a diskette.

**Note:** This warning is issued between jobs and job steps. If the extent limits are exceeded before the job or job step ends, this job is terminated.

The RCPCH parameter is not allowed in a dynamic partition.

### RCZERO

Re-initializes the return code of the last job step (as the JOB or /& statement would).

### RF=YES | CREATE

Defines the status of the recorder file (IJSYSRC) on SYSREC. It may be specified only after IPL and before the first JOB statement.

#### YES

Indicates that an active recorder file exists. The system opens this file when the first JOB statement is encountered.

#### CREATE

Instructs the system to create a recorder file when the first JOB statement is encountered.

The RF operand is not allowed in a dynamic partition.

### SDL

This operand must be specified as the last operand of the SET command. It indicates that phase names are to be added to the system directory list and, optionally, that phases are to be loaded into the SVA, including phases that are to be moved from the SVA to the logical transient area in order to be executed.

The predefined VSE ASI (automated system initialization) procedure \$0JCL includes all SET SDL statements required to start your system. You can use the VSE skeleton SKJCL0 if you want to modify this procedure. For details refer to "JCL Startup Procedures and Jobs" and "Skeletons for Starting Up BG Partition" in the manual *z/VSE Administration*.

SET SDL may be issued at any time after IPL. If several SET SDL commands are entered, the new phases specified are added to those already in the SDL. Duplicate phase names within one SET SDL command are ignored. An existing entry is replaced only if a following SET SDL command specifies the same phase name as an earlier command. Note that in this case a fresh copy of the phase is loaded each time a SET SDL command for that phase is issued; multiple specifications may thus lead to an 'SVA full' condition.

The loading and especially the copying of a phase with SVAPFIX should be carefully evaluated, since the corresponding real storage is removed from the page pool.

To build the SDL, job control reads the names of the phases which are to go into the SDL. The system searches for the requested phase in the active LIBDEF PHASE,SEARCH chain, if any, and in the system sublibrary IJSYSRS.SYSLIB. If it does not find this phase, it creates a dummy entry. This is filled when a phase is cataloged with that name.

If you want to create an SDL entry for a non-SVA-eligible phase, this phase must be in IJSYSRS.SYSLIB.

If the SET SDL command is entered from the console, the operator is prompted for the phase names. If the command is entered from SYSRDR, the phase

names must be on SYSIPT. This implies that, if the SET SDL command and the phase names are in a JCL procedure, this procedure must be cataloged with the operand DATA=YES in the librarian CATALOG command.

The phase names must be specified in one of the following formats:

1. phasename[,SVA]

The operand SVA takes effect only if the named phase is SVA-eligible. It indicates that the phase itself is to be placed in the shared virtual area, in addition to having an entry inserted in the SDL. For phases in private sublibraries, FETCH/LOAD performance improvements are only achieved if the phase is loaded into the SVA; therefore, always specify SVA.

2. phasename,MOVE

This format is valid only for B- or C-transient phases. MOVE indicates that the specified phase is to be loaded into the SVA in order to be moved from there to the respective transient area when the phase is to be executed. A phase specified with MOVE must be self-relocatable.

3. phasename,INACT

INACT indicates that the SDL entry of phasename must be flagged inactive. Neither the SDL entry (if any), nor the corresponding SVA phase (if any) are actually deleted. Any subsequent FETCH/LOAD/CDLOAD macro request for phasename will load phasename from the appropriate sublibrary in the LIBDEF PHASE chain.

**Notes:**

- a. The INACT attribute is meant for private SVA phases only. For the sake of system integrity you must not inactivate SDL entries pertaining to system or OEM phases. Especially do not inactivate SDL entries pertaining to JCL exit routines (\$JOBEXIT or \$JOBEXITn).
- b. It is not recommended to mix the INACT attribute with other attributes (SVA, MOVE or none). However, if attributes are mixed then INACT entries are processed first and not according to the sequence of occurrence. For example:

```
SET SDL
  TEST1,SVA
  TEST2,INACT
  TEST3,SVA
/*
```

This will first inactivate the SDL entry of TEST2 (if any) and then create or modify SDL entries for TEST1 and TEST3.

- c. The TEST2, INACT input in the example above is processed without any error or information message, no matter whether an SDL entry for TEST2 exists or not. From a FETCH/LOAD/CDLOAD macro point of view there is no difference, whether an SDL entry for TEST2 does not exist or whether an existing SDL entry for TEST2 is flagged inactive. In both cases TEST2 will be loaded from the appropriate sublibrary in the LIBDEF PHASE chain.
- d. When the SDL is full (that is LIBR LISTDIR SDL O=STAT displays 0 FREE ENTRIES), then SET SDL processing displays message 1T10I and all submitted phasenames with attributes other than INACT are ignored. Phasenames with attribute INACT are processed, that is their SDL entries (if any) are flagged inactive.

After the last phasename, you must enter a /\* statement to indicate the end of the input.



The maximum number of SDL entries is specified during IPL with the SVA command. If the maximum number is exceeded, a message is issued and all following statements with attributes other than INACT are flushed until a /\* or /& statement is encountered.

Following the SET SDL command, you can also define a load list in the form

```
LIST=loadlistname
```

where **loadlistname** specifies the name of the list you want to be retrieved by the system. This name must conform to the naming conventions for a phase.

The system searches for the specified list in the currently active search chain for phases (as defined by a LIBDEF PHASE statement). The system handles a correct load list the same way as it handles an SVA load list during IPL: it extracts the phase names from the list and loads the phases into the SVA. For more information on load lists, refer to "Loading Phases into the SVA" in the *z/VSE Guide to System Functions*.

#### **UPSI=config**

Sets the bit configuration of the UPSI byte in the communication region. Specify one to eight characters, either 0, 1, or X. Bit positions containing 0 are set to 0; positions containing 1 are set to 1; positions containing X are unchanged. Unspecified rightmost positions are assumed to be X.

The UPSI byte is reset to zero by a JOB or /& statement.

## SETDF (Set 3800 Printer Defaults)

The SETDF command allows the operator to set and/or reset default values for the IBM 3800 Printing Subsystem or to display the default values. The command is valid only for a 3800. The following values can be defaulted:

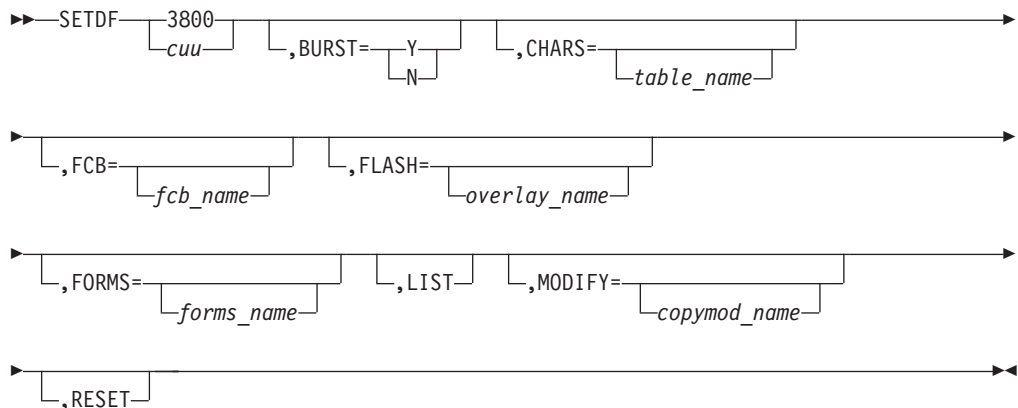
- Bursting or continuous forms stacking
- One character arrangement table
- The forms control buffer name
- The forms overlay name
- The paper forms identifier
- The copy modification name
- The setting of all hardware defaults with one command.

For further information on the 3800 and on the various ways that you can use its defaults, see the *IBM 3800 Printing Subsystem Programmer's Guide*

The length of the SETDF operator command is limited to one line of 71 characters. However, defaults are retained from one command to another (that is, if CHARS is set by one command and the next command sets FCB, then both are now set). Coding 'keyword=,' for an individual parameter makes the hardware default effective only for the specified keyword.

Issuing the SETDF command does not change job or program originated settings of the 3800. Instead, the parameters are saved such that when a user specifies DFLT=Y or keyword=\* in a SETPRT job control statement or SETPRT macro, the SETPRT routine sets the predefined defaults.

### AR Format



#### 3800

Specifies that all 3800 printers will be set with the specified default values of SETDF, or (if LIST is specified) the defaults for all of these printers will be displayed.

#### cuu

Specifies the device number of the 3800 whose default values are to be set or displayed by SETDF.

#### BURST=,

No change in the threading of the forms is requested.

**BURST=Y** specifies that the printed output is to be burst into separate sheets with the edges trimmed.

**BURST=N** specifies that the printed output is to be in continuous fanfold mode. If BURST has not been specified since the system was initialized, BURST=N is assumed.

**CHARS=,**

The default for the character arrangement table is reset to the hardware default Gothic-10 folded table.

**table-name** specifies the 1- to 4-character suffix of the name of the default character arrangement table. Only the first character arrangement table can be defaulted; multiple table names are not allowed.

**FCB=,**

The default for the forms control buffer is reset to the hardware default FCB.

**fcf-name** specifies the 1- to 4-character suffix of the name of the default FCB.

**FLASH=,**

No flashing is done.

**overlay-name** specifies the 1- to 4-character name of the forms overlay frame to be used as the default.

**FORMS=,**

The operator is requested to load the forms named STANDARD when the default is needed.

**forms-name** specifies the 1- to 4-character name of the forms to be used.

**LIST**

Specifies that the established default settings are to be displayed at the operator console. If blanks are shown for the value of a displayed keyword, this indicates the hardware default (with the exception of the BURST keyword. The default for BURST is indicated by an N.)

**MODIFY=,**

No copy modification is done.

**copymod-name** specifies the 1- to 4-character suffix of the name of the modification phase to be loaded from the library into the 3800.

**RESET**

Sets all keywords to the hardware defaults, which are:

- BURST=N
- A Gothic 10-pitch folded character arrangement table
- A 6-lines-per-inch FCB with channel-1 code on the first printable line, no other channel codes, and the forms length determined by the paper loaded;
- No forms overlay flashing
- No specific forms requested
- No copy modification done.

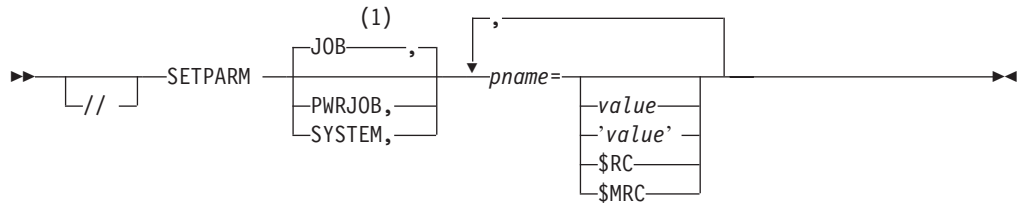
## SETPARM (Set Symbolic Parameter)

The SETPARM statement enables you to define a symbolic parameter and/or assign a value to it. This value can then be tested in an IF statement, or used by job control in subsequent statements.

If, for example, you code PARM1=SYS001 for pname=value, the symbolic parameter &PARM1 in a subsequent statement will be substituted with the value 'SYS001' by job control.

Continuation lines are allowed for this statement.

### JCC, JCS Format



#### Notes:

- 1 SETPARM JOB is only accepted within a job.

#### JOB

Specifies symbolic parameters at level n, which are valid for the (DOS) job or cataloged procedure currently active. They will be cleared at End-of-Job time (processing of /&) or End-of-Procedure time (processing of /+).

#### PWRJOB

Specifies symbolic parameters at POWER job level, which are valid for the POWER job currently active. They will be cleared at POWER EOJ time (processing of \* \$\$ EOJ). It makes no difference if you specify SETPARM PWRJOB in a job or in a cataloged procedure.

#### SYSTEM

Specifies symbolic parameters at system level, which are valid for all partitions during the lifetime of the system. They will be cleared when the system is shutdown or (re)-IPLed. It makes no difference if you specify SETPARM SYSTEM in a job or in a cataloged procedure.

#### pname

Is the name of the symbolic parameter which you want to define, or to which a value is to be assigned. It may consist of 1 to 7 alphanumeric characters, of which the first must be alphabetic.

#### value

Specifies a character string of up to 50 characters. If the string contains national or special characters, it must be enclosed in quotes. For detailed information on the format of strings assigned to parameters, see "Symbolic Parameters" on page 51. You can specify a null string as the value of a symbolic parameter, either by omitting value, \$RC and \$MRC, or by coding two quotes ( ' ') in place of the character string.

#### \$RC

Specifies the return code of the last job step which was executed. It is assigned to the parameter as a string of four characters.

**\$MRC**

Specifies the maximum return code of all preceding job steps. It is assigned to the parameter as a string of four characters.

**Notes:**

1. The PROC, EXEC PROC and EXEC REXX statements apply to symbolic parameters at level n. They do **not** apply to symbolic parameters at POWER job level or at system level.
2. Symbolic parameters are substituted according to the following search sequence:
  - a. symbolic parameters at level n
  - b. symbolic parameters at POWER job level
  - c. symbolic parameters at system level
3. Symbolic parameters at POWER job level or at system level can be displayed with the QUERY SETPARM command. See “QUERY (Query Data Spaces and Standard Options)” on page 186 for details.

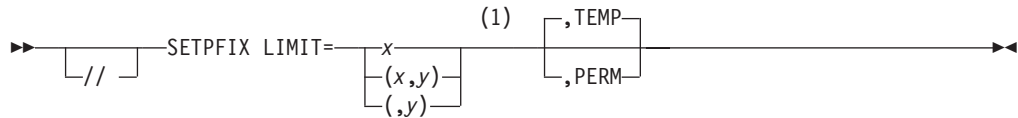
## SETPFIX (Set PFIX Limits)

The SETPFIX statement enables you to define guaranteed limits for PFIXing pages. These limits are set for the partition in which SETPFIX is issued. Two limits can be specified: one for pages to be PFIXed in page frames below the 16MB line and one for pages to be PFIXed in page frames above the 16MB line. A program which then executes in the partition can be sure that pages can be PFIXed up to these limits. The PFIX limits can either be set till end-of-job (with the TEMP operand) or beyond end-of-job (with the PERM operand).

SETPFIX is supported in static **and dynamic** partitions. For static partitions, consider the SETPFIX statement as a replacement for the ALLOC R command. Whenever a program needs real storage to PFIX pages, the SETPFIX statement should be used. ALLOC R should be used only when real execution of programs (with EXEC program,REAL) is required.

Note that an ALLOC R specification and a PFIX limit for page frames below the 16MB line are mutually exclusive, that is, when real storage is already allocated to a static partition (with ALLOC R), no PFIX limit (BELOW) can be set for this partition, and vice versa. However, ALLOC R and a PFIX limit for page frames ABOVE the 16MB line is possible for this partition, and vice versa.

### JCC, JCS Format



#### Notes:

1 'x' and 'y' must be specified as nK or mM.

**LIMIT=x | (x,y) | (,y)**

Defines the maximum amount of storage that can be PFIXed by a program running in the current partition:

- x is the maximum that can be PFIXed in page frames BELOW the 16MB line,
- y is the maximum that can be PFIXed in page frames ABOVE the 16MB line.

x and y must be specified as nK or mM, where n must be a multiple of 4 (otherwise it is rounded to the next multiple of 4).

If only one limit is specified, the other limit remains unchanged. If both limits have been specified and one limit cannot be set by the system (because, for example, there are no page frames available above the 16MB line or an ALLOC R has already been issued), the other limit is not set either.

The maximum amount which can be specified (for one limit) is 127MB or 131,068KB. The accepted value depends, of course, on the actually available page frames in the two PFIX areas (below and above the 16MB line). The MAP REAL command can be used to find out how many page frames are available.

The sum of the PFIX limit below the 16MB line (or the amount of real storage allocated to the partition via ALLOC R) and the PFIX limit above the 16MB line must not exceed the virtual size of the partition.

A specification of 0K (or 0M) resets the PFIX limit(s).

#### **TEMP | PERM**

Specifies the duration of the defined PFIX limits.

TEMP overrides all previous settings of the PFIX limits for the duration of the current job. At end-of-job, the previously defined permanent limits become effective.

**Note:** If you define a temporary limit that is smaller than the corresponding permanent limit, the permanent limit remains in effect.

PERM causes all previous settings of the specified PFIX limit(s) to be overridden permanently. The specified permanent limits are not reset at end-of-job; they remain valid until the partition is UNBATCHed or deallocated (in the case of a dynamic partition).

If only one of the two limits is specified - LIMIT=x or LIMIT=(,y) - the other (previously specified) PERM and/or TEMP limit remains in effect.

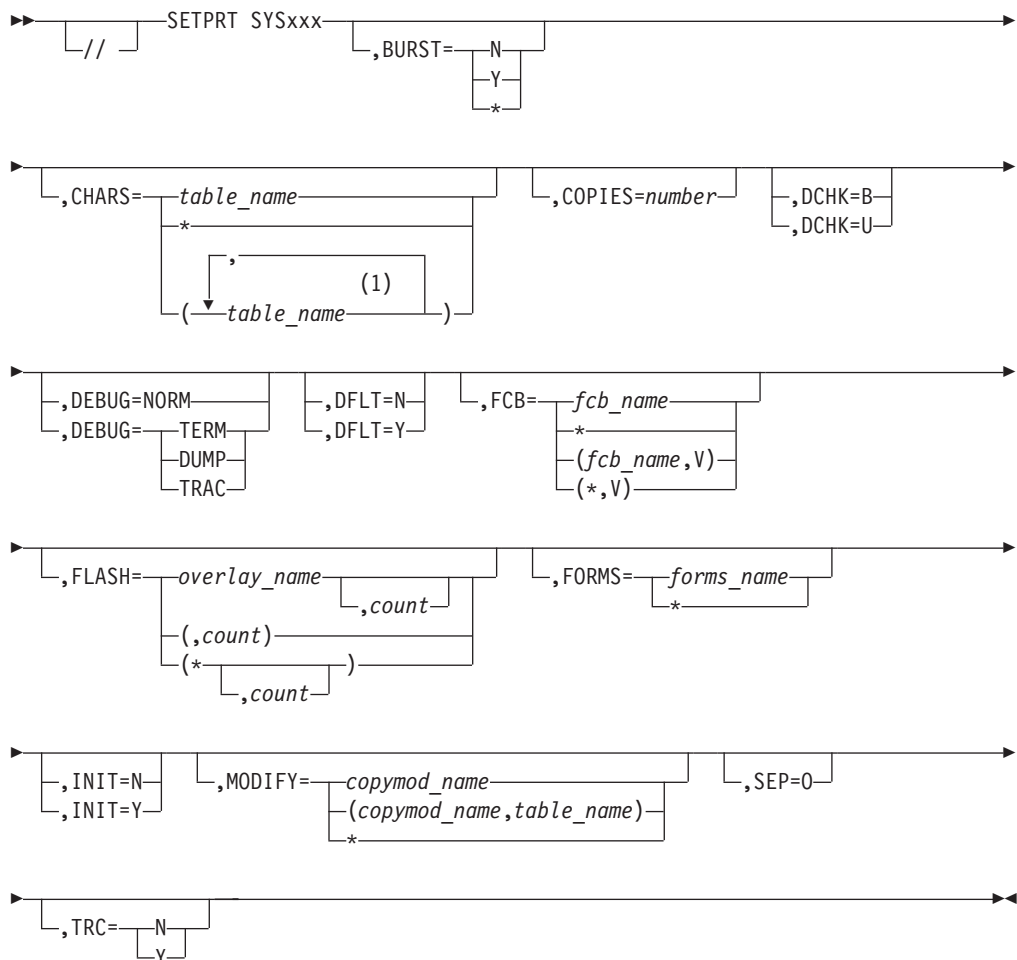
In the MAP command, only the currently effective limits are displayed.

## SETPRT (Set 3800 Printer Values for Job)

The SETPRT job control statement or command sets user-specified control values for the IBM 3800 Printing Subsystem. These values are reset at the end of the current job to the installation's default values as specified in the SETDF attention routine command, or to the hardware defaults if SETDF has not been issued. For more information on the 3800 and its use, see the *IBM 3800 Printing Subsystem Programmer's Guide*.

At least one of the optional operands must be specified. Continuation lines are accepted for the SETPRT statement.

### JCC, JCS Format



#### Notes:

- 1 Up to four names can be specified.

#### SYSxxx

Logical unit identifier for the 3800 printer to be set up. This operand is always required. SYSxxx can be SYSLST or SYSnnn. The logical unit must have been previously assigned to a 3800.

#### BURST=

If the operand is omitted, no change to the threading is requested.



**Y** specifies that the operator should thread the forms through the Burst-Trimmed-Stacker.

**N** specifies that the operator should thread the forms to the continuous forms stacker.

**\*** specifies that the system default BURST setting is requested.

**CHARS=**

If the operand is omitted, the character arrangement table is not changed unless INIT=Y is coded.

**table-name** specifies the 1- to 4-character suffix of the name of the character arrangement table.

**(table-name,...)** specifies up to four names, separated by commas and enclosed in parentheses. However, see Note under the MODIFY operand. Embedded null values, such as CHARS=(AA,,BB) or CHARS=(,AA), are not allowed. For the names of the IBM-supplied character arrangement tables, see the *IBM 3800 Printing Subsystem Programmer's Guide*.

**\*** specifies that the system default character arrangement table is requested. If the operator has not specified a default for CHARS, the hardware default Gothic-10 folded table is used.

**COPIES=**

If the operand is omitted, the number of copies is not changed unless INIT=Y is coded.

**n** specifies the number of copies of each page to be reproduced before printing the next page. It can be a value from 1 to 255.

**DCHK=**

If the operand is omitted, data checks are blocked.

**B** specifies that data checks are to be blocked. This means that unprintable characters in the data transmitted to the 3800 are printed as blanks.

**U** specifies that data checks are allowed.

**DEBUG=**

**NORM** sets a return code in register 15 and returns to the caller on any exit from the SETPRT routines. This is in effect if DEBUG is omitted from all preceding SETPRTs in the job.

**TERM** sets a return code in register 15 and cancels the activity for return codes higher than 4. For return codes 0 and 4, TERM has the same effect as NORM.

**DUMP** sets a return code in register 15 and cancels the job with a dump, for a return code higher than 4.

**TRAC** dynamically traces, on SYSLST, the activity of the SETPRT routines and then cancels the job with a dump if the SETPRT return code is greater than 4. Tracing requires 12K of GETVIS space.

**DFLT=**

**N** is the default specification for this keyword and does not establish 3800 default setup.

**Y** specifies that the printer is to be set with the defaults that were specified by the operator in the SETDF command. It is equivalent to coding **\*** for each of the operands BURST, CHARS, FCB, FLASH, FORMS, and MODIFY that are not specified.

**FCB=**

If the FCB operand is omitted, the FCB is not changed unless INIT=Y is coded.

**fcname** specifies the 1- to 4-character suffix of the name of the FCB. The length of the form defined by FCB must match the length of the form loaded in the 3800, as specified with the FORMS operand.

V requests FCB verification. The FCB contents are formatted and printed on the 3800. Data checks are blocked, and translate table zero is used for printing the FCB verification page.

\* specifies that the system default FCB is requested. If the operator has not specified a default FCB, the hardware default FCB is 6 lines per inch with a channel-1 code defined on the first printable line, and the length set equal to that of the form currently loaded.

**FLASH=**

**overlay-name** is the 1- to 4-character name of the forms overlay frame that the operator will be requested to insert in the 3800.

**count** is the number (from 0 to 255) of copies to be flashed with the overlay, beginning with the first copy of the first transmission. If 0 is specified, the specified forms overlay frame is mounted or remains mounted but is not flashed. A specification of FLASH=(,100), for example means to flash the current forms overlay frame for 100 copies.

If no count is specified, all copies are flashed.

\* requests the system default forms overlay. If the operator has not specified a default, no flashing occurs.

**FORMS=**

**forms-name** is the 1- to 4-character forms identifier. If the specified forms are not already loaded, a message to the operator requests the specified forms. If the new form has a length different from the previous form and a new FCB is not specified, the 3800 loads the hardware default FCB. This can cause erroneous results later. To avoid this problem, specify a new FCB when loading forms of a new length.

\* requests the system default forms. If the operator has not specified a FORMS default, form STANDARD is requested.

**INIT=**

Y specifies that the printer be reset to hardware defaults of a 6-lines-per-inch FCB with channel-1 code in the first printable line, a Gothic-10 folded character arrangement table, one copy, and no flashing of forms overlays. Copy modification is cleared, and burster threading, forms, and blocking or unblocking of data checks are not changed. If TRC=Y is not also coded, then lines written to printer files opened after this SETPRT should not contain table reference characters unless such an inclusion is specified in the DTFxx macro. The TRC indicators for an open printer file are not changed. (Any of these actions can be overridden with other keywords.)

N is the default and does not reset the 3800 to hardware defaults.

**MODIFY=**

If the operand is omitted, then the currently-loaded copy modification phase is used, unless INIT=Y is also coded.

**copymod-name** specifies the 1- to 4-character suffix of the name of the copy modification phase that was assigned when the phase was built.

**table-name** specifies the 1- to 4-character name of the character arrangement table to be used when the 3800 prints the copy modification text. This character arrangement table need not be one of those specified with the CHARS operand. However, see Note below. If table-name is omitted, the first character arrangement table specified with the CHARS operand (or the default, if none is specified) is used.

\* requests the system default copy modification. If the operator has not specified a MODIFY default, any existing copy modification is eliminated.

**SEP=**

Omission of the SEP operand indicates that no data set separation is required.

**O** indicates that, if the burster-trimmer-stacker is being used, the 3800 should offset-stack the pages that follow from the pages that were previously transmitted. If the continuous forms stacker is being used, the 3800 changes the marking on the perforation edge from one line to two lines or vice versa.

**TRC=**

**N** indicates that, for any DTFPR or DTFDI operand after this SETPRT, data lines do not contain table reference characters unless specified in the DTF macro. The table reference character will not be prefixed to each data line when presented to the access method.

**Y** indicates that the first character of each output data line (after the optional print control character) given to the access method is a table reference character. This applies only to PUT macros with DTFPR or DTFDI.

**Note:** The total number of character sets referenced by character arrangement tables in both the CHARS and MODIFY operands cannot exceed the number of writable character generation modules available on the 3800 (either two or four). If a character set is referenced by multiple character arrangement tables and graphic character modification is not used, then only one copy of that character set is loaded into the 3800. If a character set is referenced by two character arrangement tables and one is modified by graphic character modification and the other is not, then two character sets are loaded.

## SETPRT

### Example

The following example shows the use of the SETPRT job control statement to set up the 3800 Printing Subsystem with the physical unit address 118:

```
// JOB D63SETP          SET UP 3800 Printer
// ASSGN SYS010,118     ASSIGN SYS010 TO 3800 PRINTER
// SETPRT SYS010,BURST=Y,DCHK=B,SEP=0,TRC=Y,          C
  FORMS=X,FLASH=(TEST,2),FCB=(STD6,V),              C
  CHAR=(X,XX,XXX,GF12),MODIFY=(CMO1,FM12),COPIES=4
/&
```

The operands of the SETPRT statement specify:

- BURST=Y specifies that the operator will be asked to thread the forms through the Burster-Trimmed-Stacker.
- DCHK=B specifies that data checks are to be blocked.
- SEP=0 specifies that the burst pages from this job are to be offset in the stacker from those of the previous job.
- TRC=Y specifies that the first character of each output data line (following the optional print control character) is a table reference character.
- FORMS=X specifies that the forms named X are to be used for printing this job.
- FLASH=(TEST,2) specifies that the first 2 copies of each page printed are to be flashed with the forms overlay named TEST.
- FCB=(STD6,V) specifies that the forms control buffer phase named STD6 is to be used, and that the FCB contents are to be formatted and printed for verification by the operator.
- CHARS=(X,XX,XXX,GF12) specifies the names of the four character arrangement tables that are to be loaded into the 3800.
- MODIFY=(CMO1,FM12) specifies that the FM12 character arrangement table, which uses Format 12-pitch characters, is to be used to print the copy modification named CMO1.
- COPIES=4 specifies that 4 copies of each page of the file are to be printed in a group before printing 4 copies of the next page.

## SIZE (Program Size)

The SIZE command is used to specify the amount of contiguous virtual storage in a partition which is reserved for program execution. The rest of the partition is available as partition GETVIS area.

The SIZE command is not allowed in a dynamic partition.

The SIZE command has a function similar to the SIZE operand of the EXEC statement. The difference is that:

- The SIZE **command** makes a permanent change which lasts until another SIZE command is issued, or until the next IPL.
- The SIZE **operand** of the EXEC statement is effective only for the current job step.

The SIZE operand of the EXEC statement is still effective for its own job step after a SIZE command has been issued.

If a program running in real mode needs GETVIS space, the SIZE operand of the EXEC statement has to be specified. The SIZE command does not provide GETVIS space for a program running in real mode.

The SIZE command is not accepted for an active partition which is using its GETVIS space.

### AR, JCC Format



#### BG | Fn

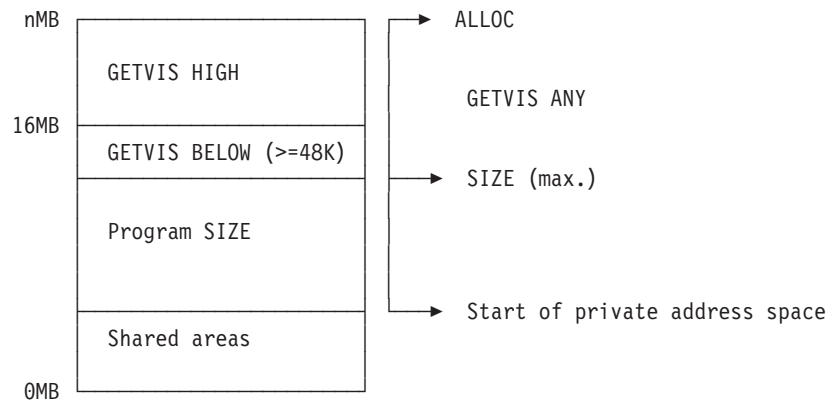
Indicates the static partition (BG, F1, F2, ...) for which storage is to be reserved.

#### nK | mM

Specifies the amount of storage to be reserved for program execution in kilobytes (nK) or megabytes (mM). The remainder of the partition is available as partition GETVIS area. **n** should be a multiple of 4. If not, the system rounds it up to the next higher multiple of 4. The minimum value is 80K.

The maximum permissible value is the partition size (as specified by ALLOC) minus the minimum partition GETVIS area, which is 48K. Since the SIZE definition must not cross the 16MB line, the system ensures that the start of the partition GETVIS area is not moved beyond the (16MB minus 48KB)-line:

## SIZE



$$\text{SIZE(max)} = 16\text{MB} - 48\text{KB}(\text{min. GETVIS BELOW}) - \text{size of shared areas}$$

## START (Start or Continue Processing)

The AR START command activates or continues processing in the specified static partition.

The function of the START command is exactly the same as that of the BATCH command.

The JCC START command can only be used to start a partition which has not yet completed its ASI (Automated System Initialization) job control procedure. It is not allowed in a dynamic partition.

### AR Format



### JCC Format



### BG

Indicates that the background partition is to be reactivated.

**Fn** Indicates that the specified foreground partition is to be activated, or restarted after having been stopped by a STOP command.

If the operand is omitted in the AR command, BG is assumed.

## STATUS (Display Task or Device Status)

The STATUS command can be used to inspect the status of all active tasks in the system or of a certain device. It provides appropriate information about all possible types of 'bound' conditions on the operator console and is intended to assist the system operator in making the correct decision in case of any problems that he may have encountered. The command cannot be used in a job stream.

To retrieve task-related information, issue the STATUS command in the following format:

### AR Format



### BG | Fn

Specifies the partition (BG, F1,...) that the operator wants to inspect for its current status.

### jobname

Indicates the job name of the job that the operator wants to inspect for its current status. *jobname* can be up to 8 characters and must be unique.

### SYS

Specifies that the system task status is to be retrieved.

If neither **partition** nor **SYS** has been specified, the status of all tasks which are currently active will be displayed.

The following snap from the operator console is used to explain the task-related information retrieved with the command:

```

1. 01 STATUS
2. 02 AR 015   PMR->F4   WAITING FOR I/O ON DEVICE=160
3. 03 AR 015   DIR->F5   WAITING FOR I/O ON DEVICE=161
4. 04 AR 015   CST->AR   WAITING FOR I/O ON DEVICE=170
5. 05 AR 015   AR       READY TO RUN
6. 06 AR 015   S01 -F4   WAITING FOR PAGE I/O COMPLETION
7. 07 AR 015   S02 -F4   READY TO RUN
8. 08 AR 015   S03 -F4   WAITING FOR I/O, ECB OR TECB
9. 09 AR 015   F4       WAITING FOR I/O, ECB OR TECB
10. 10 AR 015   F5       WAITING FOR PROGRAM FETCH
    11 AR 015 1I40I   READY
    12
    13

```

1. This is the command as entered by the operator.
2. The page manager (PMR) system task, currently working for partition F4, is waiting for its I/O operation on device 160 to complete.
3. The directory (DIR) system task, currently working for partition F5, is waiting for its I/O operation on device 161 to complete.
4. The console (CST) system task, currently working for the AR partition, is waiting for its I/O operation on device 170 to complete.

Other system tasks that can appear in the listing are:

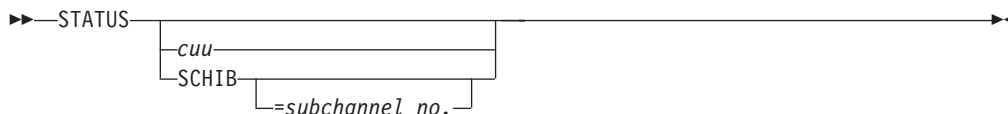


ASY Asynchronous console processing  
 DSK Resident disk error recording  
 ERP Transient error recording  
 FCH Fetch  
 HCF Hard Copy File  
 LCK Lock manager  
 LOG Logging task  
 PGN Page-in  
 RAS Reliability, Availability, Serviceability  
 SNS Sense  
 SPT Service Processor Task  
 SUP Supervisor fetch  
 SVT Service task

5. The AR partition is ready to run. (The AR partition does not have to wait for CST since its I/O operation has been console-buffered.)
6. S01, subtask 1 attached by partition F4, is waiting on a page I/O request started by the PMR system task (see 2) to be completed.
7. S02, subtask 2 attached by partition F4, is ready to run.
8. S03, subtask 3 attached by partition F4, is waiting on either an I/O, ECB or a TECB; no further distinction was possible.
9. The main task of partition F4 is waiting on either an I/O, ECB or a TECB; no further distinction was possible.
10. The main task of partition F5 is waiting for program fetch to complete its function (see 3).

To retrieve device-related information, issue the STATUS command in the following format:

### AR Format



The response to this command is a header line followed by a line of information for a single subchannel or for all subchannels in the system. The header line identifies, column by column, the SCHIB information below it.

#### cuu

Specifies the address of the device that the operator wants to inspect for its current state. The subchannel related to this device will be displayed in a formatted way, assuming a valid subchannel number exists for this device (see SCHIB description).

#### SCHIB[=subchannel-number]

SCHIB causes the SubCHannel Information Block of all subchannels defined for this system to be displayed in a formatted way. A subchannel number (=subchannel-number) can be appended to this operand to restrict the output to a single subchannel.

The following fields will be returned:

DEV Device number  
 INT-PARM Interrupt parameter  
 ISC Interruption subclass code  
 FLG Flag field  
 LP Logical Path Mask  
 PNO Path Not Operatinal Mask

## STATUS

LPU	Last Path Used Mask
PI	Path Installed Mask
MBI	Measurement Block Index
PO	Path Operational Mask
PA	Path Available Mask
CHPID0-3	Channel Path Identifiers 0 through 3
CHPID4-7	Channel Path Identifiers 4 through 7

If the device is currently active, the following additional information will be provided:

KEY	SCSW protect key
SLCC	Logout condition code bits
FP1AUZEN	Various bits from the SCSW (bits 8-15)
FCTL	Function control bits
ACTL	Activity control bits
SCTL	Subchannel control bits
CCW-ADDR	SCSW CCW address
DS	SCSW device status bits
CS	Subchannel status bits
CNT	SCSW residual byte count

## STDOPT (Standard JC Options)

The STDOPT command or statement sets or resets the permanent job control options which were established at system initialization (system defaults). The permanent options established are identical with the default values of the STDOPT command. If no STDOPT command is given, all the default values are valid. The STDOPT command can be given in any partition, but the values specified apply to **all** partitions. To be active for a dynamic partition, however, the options must be set before the dynamic partition is started.

If an option is reset, its new value becomes effective in a static partition after the next /& or JOB statement is issued in that partition. (Exceptions: LINES becomes effective immediately, DATE can become effective earlier in a partition if a GETIME macro is issued in that partition.)

An option specified with STDOPT can be **temporarily** overridden in one partition by the OPTION statement. (Exception: LINES can only be overridden by the SET LINECT command; DATE cannot be overridden.)

The QUERY STDOPT command displays the current setting of permanent job control options.

### JCC, JCS Format



The options, which can appear in any order, are as follows (the first specification is always the system default value):

#### ACANCEL=NO | YES

Specifies whether job control is to cancel jobs automatically (ACANCEL=YES) or to wait for operator intervention (ACANCEL=NO) after an unsuccessful attempt to assign a device. (Note that the LOG command overrides an ACANCEL=YES specification.)

#### ACL=YES | NO

ACL=YES indicates that in case of multi-volume files and automatic cartridge loading being active on the actual device, the access method will process all tapes (cartridges) on the actual device first and then follow the alternate chain. (Normal ACL processing.)

ACL=NO specifies that in case of multi-volume files, the access method will follow the alternate chain, independent of whether automatic cartridge loading is active on the device or not.

#### ALIGN=YES | NO

Specifies whether the High Level Assembler for VSE is to align data on halfword or fullword boundaries, according to the type of instruction used.

#### CHARSET=60C | 48C

Specifies either the 48- or 60-character set for PL/I translator input on SYSIPT.

#### DATE=MDY | DMY

Specifies the format of the date:

MDY=month/day/year

DMY=day/month/year.

**Note:** Changing the date format without reformatting the VSE/POWER spool files at the same time will result in incorrect interpretation of the creation date of existing VSE/POWER queue entries. For details and recommendations refer to “Date Recording and Date Format” in the *VSE/POWER Administration and Operation* manual, SC33-8247.

**DECK=**YES | NO

Specifies whether or not language translators are to produce object modules on SYSPCH.

**DSPDUMP=**NO | YES

Specifies whether a dump of the data spaces is to be taken in the case of an abnormal program end. For details see // OPTION DSPDUMP.

**DUMP=**YES | NO | PART

Specifies whether or not a dump of the registers and virtual storage is to be taken in the case of an abnormal program end. PART specifies that a dump of the major supervisor control blocks and the virtual storage of the partition is to be taken. The specification of DUMP=NO suppresses the DSPDUMP=YES option.

**ERRS=**YES | NO

Specifies whether or not language translators are to summarize all errors in source programs on SYSLST. Assembler and PL/I always assume ERRS=YES.

**HCTRAN=**YES | NO

Specifies whether the output from PRINTLOG and LISTLOG is to be translated to all uppercase (YES) or in mixed, upper- and lowercase. Default is YES.

**JCANCEL=**NO | YES

Specifies whether the system should skip to end-of-job when a job control error occurs (JCANCEL=YES), or wait for operator intervention (JCANCEL=NO).

**LINES=**56 | nn

Specifies the number of lines per page on SYSLST. The minimum is 30, the maximum is 160. (If job control is running in another partition at the same time, the new value becomes effective in that partition when the next page is started.)

**LIST=**YES | NO

Specifies whether or not language translators are to write source module listings and diagnostics to SYSLST.

**LISTX=**NO | YES

Specifies whether or not language translators are to write hexadecimal object module listings on SYSLST.

**LOG=**YES | NO

Specifies whether or not all job control statements are to be listed on SYSLST. Invalid statements and commands will be listed on SYSLST in any case if it is assigned.

**RLD=**NO | YES

Specifies whether or not the relocation dictionary information is to be printed.

**SADUMP=**n | ([n],m)

Specifies the priority in which the partitions (and any owned data spaces) should be included in a stand-alone dump. This priority applies to **all** partitions/data spaces in the system unless it is overridden by a corresponding // OPTION statement. SADUMP=n indicates the priority of the partitions; SADUMP=(n),m indicates the priority (n) of the partitions, if specified, and the priority of any owned data spaces (m).

The values for n and m can be 0 to 9, with 9 being the highest priority and 0 indicating that no dump is needed. The IBM supplied default for both n and m is 0.

**SCANCEL=NO | YES**

This option causes an operator cancel condition to be simulated whenever a job control error cancels a program without waiting for operator intervention, for example, if OPTION JCANCEL or ACANCEL and NOLOG is in effect. This allows your program to specify a conditional ON \$CANCEL command to avoid that the job ending looks like a normal end-of-job.

**SXREF=NO | YES**

Specifies whether the assembler is to print short cross-reference lists on SYSLST. The printing of unreferenced labels is suppressed. Do not specify SXREF together with XREF.

**SYM=NO | YES**

SYM=YES specifies that the PL/I compiler is to produce a symbol and offset table listing on SYSLST, or that the COBOL compiler is to produce a data division glossary.

**SYSDUMP=NO | YES**

YES indicates that dumps are to be written to the dump sublibrary which is active for the partition. The dump sublibrary must have been defined with the LIBDEF DUMP command. SYSDUMP=NO specifies that dumps are to be written to SYSLST. For compatibility reasons, the keyword may be entered as SYSDMP.

**SYSDUMPC=NO | YES**

NO has no effect on dump processing.

YES indicates that the dump is ignored when the following two conditions are met:

- the dumps are to be written to the dump sublibrary and not to SYSLST (SYSDUMP=YES was specified)
- the dump sublibrary is full, in error or not defined.

To avoid the dump being written to SYSLST, you must specify both **SYSDUMP=YES** and **SYSDUMPC=YES**. Note that if you specified **SYSDUMP=NO**, **SYSDUMPC=YES** has no effect.

**TERM=NO | YES**

Specifies whether messages from a compiler are to be displayed on SYSLOG.

**XREF=YES | NO**

XREF=YES specifies that the assembler is to write symbolic cross-reference lists on SYSLST, or that American National Standard COBOL is to produce a cross-reference listing.

Do not specify SXREF together with XREF. If you specify XREF=YES|NO the SXREF operand defaults to NO.

## STOP

### STOP (Stop Processing)

The STOP command indicates that there are no more jobs to be executed in the partition in which the command is given.

The command is not allowed in a dynamic partition.

#### JCC Format

▶▶—STOP—▶▶

The STOP command has no operand.

This command removes the partition from the system's task selection mechanism, but the partition remains active. Job control remains in the partition and can be restarted by the START or BATCH attention routine command.

## SYSDEF (Define Data Space)

The SYSDEF command is restricted to the attention routine (AR) and to the BG partition.

The SYSDEF DSPSPACE command defines the following:

- Limits and defaults for data spaces (DSPSPACE).
- Starts and stops CPUs and resets Turbo Dispatcher information (see “QUERY TD” on page 192 for details).
- SCSI devices (see “QUERY SCSI” on page 191 for details).
- The total amount of virtual storage which may be allocated to data spaces (this storage is taken from the IPL VSIZE).
- The maximum number of data spaces which can be allocated within the system or per partition at one time.
- The maximum number of data spaces with SCOPE=COMMON which can be allocated at one time.
- The default size of a data space.

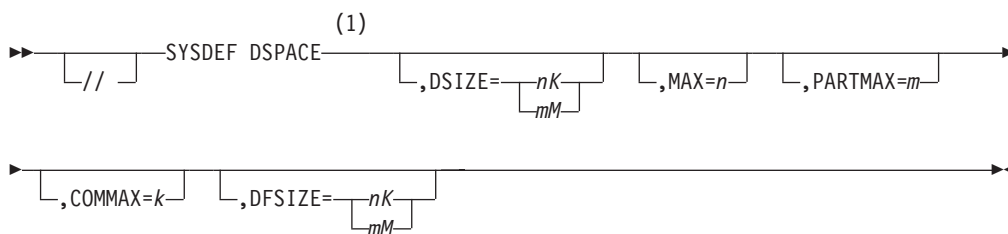
The QUERY DSPSPACE command can be used to display these limits and defaults, plus detailed information on data spaces, like data space names or sizes. See also “QUERY (Query Data Spaces and Standard Options)” on page 186.

The MAP command will display the amount of virtual storage which is allocated to data spaces.

You can also use options which control whether data spaces are to be dumped in case a program ends abnormally.

**Note:** The SYSDEF command can only be used from the BG partition and as an attention routine command.

### AR, JCC, JCS Format



#### Notes:

- 1 DSPSPACE must be the first operand, and at least **one** of the following keyword operands has to follow (in any sequence).

For all operands of the SYSDEF command, the following IBM defined initial values exist:

DSIZE=0M,MAX=256,PARTMAX=16,COMMAX=5,DFSIZEnK=960K

#### DSPACE

Indicates that data space storage is to be defined.

#### DSIZE=nK | mM

Defines the total amount of virtual storage which may be allocated (as a sum) to data spaces. Thus, DSIZE limits the virtual storage which may be occupied

by data spaces. Note, however, that the specified storage may not always be available, since it may have been allocated to partitions.

The values for nK and mM must be in the range of 0K or 0M up to VSIZE; nK must be a multiple of 32K; otherwise it is rounded up to the next multiple.

If you do not specify DSIZE, the IBM supplied default value of 0 or a previously defined value remains in effect.

DSIZE may be re-defined at any point in time. If the new DSIZE value is smaller than the amount of storage currently allocated to data spaces, the creation of new data spaces is rejected until the actual value is below the new limit (by deletion of data spaces). Currently allocated data spaces are not affected.

A specification of DSIZE=0M (or 0K) means that no more data spaces can be allocated (created).

**Note:** When defining DSIZE, the size of those data spaces that will be created for **virtual disks** has to be considered, too.

#### **MAX=n**

Defines the maximum number of data spaces which may be allocated. n must be in the range of 1 to 65535.

If you do not specify the MAX operand, the IBM supplied default value of 256 or a previously defined value remains in effect.

MAX may be re-defined at any point in time. If the new MAX value is smaller than the current number of data spaces, the creation of data spaces is rejected until the current value is below the new limit (by deletion of data spaces). Currently allocated data spaces are not affected.

#### **PARTMAX=m**

Defines the maximum number of data spaces which may be created by a single partition at any one time. m must be in the range of 1 to 512.

If you do not specify the PARTMAX operand, the IBM supplied default value of 16 or a previously defined value remains in effect.

PARTMAX may be re-defined at any point in time. If in a partition the current number of data spaces (created by the partition) is higher than the new PARTMAX value, the creation of data spaces by this partition is rejected until the current value is below the new limit (by deletion of data spaces). Currently allocated data spaces are not affected.

#### **COMMAX=k**

Defines the maximum number of data spaces with SCOPE=COMMON which may exist at any one time. (The SCOPE operand is specified in the DSPSERV assembler macro.) k must be in the range of 5 to 253 minus the number of virtual disks added at IPL time.

If you do not specify the COMMAX operand, the IBM supplied default value of 5 or a previously defined value remains in effect. The COMMAX value can be increased as long as only COMMON data spaces are within the PASNs, but it may **not** be decreased.

IBM recommends to specify COMMAX only in the JCL startup procedure \$0JCL for the BG partition.



**Notes:**

1. COMMAX does not include the number of data spaces with SCOPE=COMMON which are allocated for virtual disks (via the VDISK command).
2. For each virtual disk which has been added at IPL time (with ADD cuu,FBAV) and not overwritten by IPL device sensing, an entry is reserved in each PASN-AL (primary address space access list).
3. In addition, COMMAX entries will be reserved in the PASN-AL for data spaces with SCOPE=COMMON.

**DFSIZE=nK | mM**

Defines the default size for the creation of a single data space. The minimum default size is 32K. The specified value must be a multiple of 32K; if not, it is rounded up to the next multiple of 32K. The IBM supplied initial default value is 960K.

## SYSDEF SCSI (Define SCSI Device)

The SYSDEF SCSI command is used to associate the VSE SCSI device number (FBA) with the real SCSI Logical Unit Number (LUN), and its connection path (FCP, WWPN).

For each SCSI device a SYSDEF SCSI command, or IPL DEF SCSI command is required. In case the same SCSI device is attached via additional FCP devices (multipathing), a separate SYSDEF SCSI command is required for each path.

Each SYSDEF SCSI command causes the system to connect to the specified SCSI device. If the connection cannot be established because of an incorrectly specified configuration, the command can be reentered with corrected configuration parameters.

The SYSDEF command will reset any device down indication for the FBA cuu.

Use QUERY SCSI to query all SCSI devices defined in the system (see "QUERY SCSI" on page 191 for details).

### AR, JCC, JCS Format

(1)  
 >>> [ // ] SYSDEF SCSI ———, FBA=*cuu*—, FCP=*cuu*—, WWPN=*portname*—, LUN=*lun*————>>>

**Notes:**

- 1 Parameters can be specified in any order.

### AR Format

(1)  
 >>> SYSDEF SCSI ———, DELETE—, FBA=*cuu*————>>>  
 >>> [ , FCP=*cuu* ] [ , WWPN=*portname* ] [ , LUN=*lun* ]————>>>

**Notes:**

- 1 Parameters can be specified in any order.

**FBA=*cuu***

*cuu* is the SCSI device ADDED as FBA.

**FCP=*cuu***

*cuu* is the device number of the attaching FCP ADDED as FCP.

**WWPN=*portname***

*portname* is the 64 bit world wide port name of the SCSI controller configured to access the LUN.

It is specified in 16 hexadecimal digits. Valid specifications are 0 to 9 and A to F.

**LUN=*lun***

*lun* is the 64 bit logical unit number identifying the particular SCSI device as configured in the SCSI controller.

It is specified in 1 to 16 hexadecimal digits. Valid specifications are 0 to 9 and A to F. If less digits than 16 are specified, trailing zeros will be presumed. For example, LUN 216B0000 00000000 can be specified as LUN=216B.

#### DELETE

Indicates that a SCSI connection is to be deleted. The SCSI device specified in FBA=cuu must be set offline with the AR OFFLINE command. There must not be any ongoing I/O operations.

If only FBA=cuu is specified, then **all** paths associated with the VSE SCSI device number (FBA=cuu) are dropped. If FBA=cuu is specified together with the FCP=cuu operand, then the one path matching the specified operand values is dropped.

Each LUN can be associated with only one unique FBA cuu, and vice versa. The only exception is the multipath (MP) definition, where an FBA cuu is connected to one and the same LUN via different FCP cuu's:

```

QUERY SCSI,500
AR 0015 FBA-CUU FCP-CUU WORLDWIDE PORTNAME LOGICAL UNIT NUMBER
AR 0015 500 C00 5005076300CB93CB 5178000000000000 1
AR 0015 500MP D00 5005076300CB93CB 5178000000000000 2

```

In the above sample, FBA cuu 500 is connected to LUN 5178 via two different connection paths. The first connection path <sup>(1)</sup> is used to access the SCSI device. If access via the first connection path <sup>(1)</sup> is no longer possible, the system will switch to the next one <sup>(2)</sup>, which is displayed first in a subsequent QUERY SCSI,500 command. Thus, multipathing is used to increase the availability of SCSI-connected devices, but not for workload balancing.

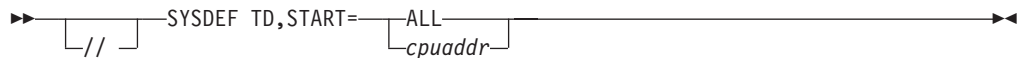
## SYSDEF TD

With the SYSDEF TD command or statement you can start and stop CPUs and reset Turbo Dispatcher information.

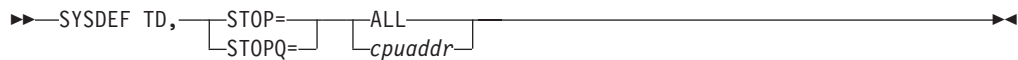
As attention routine command you can use SYSDEF TD at any time from the system console or a master console. The SYSDEF TD statement for starting CPU(s) as shown in the following syntax diagram can also be included in the startup procedure (\$0JCL) of the BG partition. Use QUERY TD to query the status of your z/VSE multiprocessor environment (see "QUERY TD" on page 192 for details).

All operands must be specified exactly in the same sequence as indicated by the syntax diagrams below.

### AR, JCC, JCS Format



### AR Format



### AR Format



**TD** Indicates that the command or statement addresses the Turbo Dispatcher.

#### START=ALL | *cpuaddr*

Initializes the multiprocessing environment and starts either all CPUs of the multiprocessor or the one CPU identified by *cpuaddr*. A CPU address can be any hexadecimal value from X'00' to X'09' (under VM/ESA<sup>®</sup> from X'00' to X'3F'). CPU activation happens at the next system check point.

A START request implies a RESETCNT request.

#### STOP=ALL | *cpuaddr*

Stops either all additionally started CPUs (except the one from which IPL was performed) or the one CPU identified by *cpuaddr*. The Turbo Dispatcher stops the CPU(s) at the next possible system checkpoint and frees all occupied resources.

A STOP request implies a RESETCNT request.

#### STOPQ=ALL | *cpuaddr*

Quiesces all additionally started CPUs or the one CPU identified at the next possible system checkpoint. A quiesced CPU is not available for task selection and will not process any work units. It will resume processing after being started via the START operand.

A STOPQ request implies a RESETCNT request.

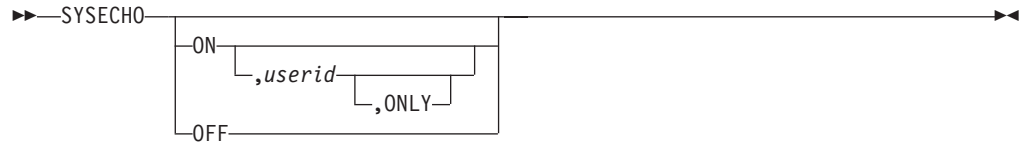
#### RESETCNT

Resets all Turbo Dispatcher related information which is displayed when a QUERY TD command or statement is given.

## SYSECHO (VM as z/VSE Master Console)

The SYSECHO command allows a VM userid to operate as a z/VSE master console.

### AR Format



If no operand is specified, the current SYSECHO settings are displayed.

#### ON | OFF

Specifies that master console routing to VM is to be activated (ON) or deactivated (OFF). The other command operands are only applicable when ON is specified. When specified with OFF, an error message is generated.

#### userid

Specifies the VM userid of the virtual machine to which messages are to be routed. This operand is required for the first SYSECHO command after IPL. When omitted for subsequent SYSECHO commands, the current userid remains in effect.

#### ONLY

Specifies that other CMS consoles are not to be supported. When this operand is specified, it remains in effect until the system is IPLed again.

The command can only be issued from a system or master console, or from the BG ASI procedure, and is rejected when the system was IPLed with the SYS option VMCF=NO.

## TLBL (Tape Label Information)

The TLBL statement contains file label information for the checking and writing of tape labels. The TLBL statement may be used with both EBCDIC and ASCII files. For more information about tape labels, refer to "Processing of User Labels" in the *z/VSE System Macros Reference*.

### JCS Format for EBCDIC Files

```

▶▶ // TLBL filename, [file_id], [date], [file_serial_number]
▶▶ , [volume_sequence_number], [file_sequence_number],
▶▶ [generation_number], [version_number] [DISP=NEW]
▶▶ , [DISP=OLD]
▶▶ [MOD]

```

### JCS Format for ASCII Files

```

▶▶ // TLBL filename, [file_id], [date], [set_identifier],
▶▶ [file_section_number], [file_sequence_number],
▶▶ [generation_number], [version_number] [DISP=NEW]
▶▶ , [DISP=OLD]
▶▶ [MOD]

```

Continuation lines are accepted for the TLBL statement.

#### filename

This can be from one to seven alphanumeric characters, the first of which must be alphabetic. This unique filename is identical to the name of the program DTF that identifies the file.

**Note:** Do not use the same filename for both a DLBL and a TLBL statement.

#### 'file-id'

One to seventeen characters contained within quotes, indicating the unique name of the file on the volume. Within the identifier, a character sequence of a quote followed by a comma or a blank is not allowed. The system interprets this sequence as the end of the identifier.

On output files, if this operand is omitted, the name specified for *filename* is used. On input files, if the operand is omitted, no checking of file identifiers will be done.

#### date

**Output files:**

The DATE operand can be omitted, or can be supplied in one of two different formats as shown below:

- Retention period format. This is specified:
  - As a decimal number nnnn (0-9999)

or, equivalently,

- as 00/nnnn (0-9999)

The nnnn is one through four decimal digits and specifies the retention period in days. You may use retention periods for new files to help reduce the chance of later accidental deletion. After the retention period, the file can be deleted or written over by another file.

Internally the system converts the retention period into an expiration date by adding up the retention period and the creation date.

- Date format. This is specified as:
  - yy/ddd with yy not equal to 00
  - 19yy/ddd
  - 20yy/ddd

The yy is a two-digit year number (00 through 99) and the ddd is a three-digit day number from 000 through 366. You may use this date format to specify the expiration date for a new file. On and after the expiration date, the file can be deleted or written over by another file.

Files with an expiration date of 1999/366 are always considered unexpired. Files with an expiration date of 1999/365 are considered unexpired, with one exception: if the expiration date 1999/365 was caused by the specification of a retention period. For example, a file created on 11/30/1999 with a retention period of 31 will expire.

The format yy/ddd will be complemented by the system to either 19yy/ddd or 20yy/ddd, dependent on the current date's year and yy. The system will complement yy/ddd to 19yy/ddd if 19yy is greater than or equal to the current date's year, and to 20yy/ddd else. For example, in 1998, the expiration date 98/ddd is complemented to 1998/ddd, whereas 97/ddd is complemented to 2097/ddd. This is because expiration dates are considered to be future-oriented rather than past-oriented.

If this operand is omitted, a 0-day retention period is assumed. The current system date is always used as the creation date for output files.

### Input files

For input files the DATE operand supplied by TLBL is compared with the actual creation date in the standard file label of the tape which is being accessed. If there is a mismatch, processing is interrupted and the system issues a message.

If the DATE operand is omitted, no checking is done for input files.

The DATE operand can be supplied in the following format:

- yy/ddd with yy not equal to 00
- 19yy/ddd
- 20yy/ddd

The yy is a two-digit year number (00 through 99) and the ddd is a three-digit day number from 000 through 366.

The format yy/ddd will be complemented by the system to either 19yy/ddd or 20yy/ddd, dependent on the current date's year and yy. The system will complement yy/ddd to 19yy/ddd if 19yy is greater than the current date's year minus 80, and to 20yy/ddd else. For example, in 1998, the DATE operand 50/ddd is complemented to 1950/ddd, whereas 05/ddd is complemented to 2005/ddd. This is because creation dates are supposed to be dates belonging rather to the past than to the future.

**file\_serial\_number (EBCDIC) or set identifier (ASCII)**

One to six characters indicating the file serial number of the first (or only) reel of the file. For input and output tapes, specify the six-digit volume serial number given to the tape reel when it was initialized.

The operand may be omitted. In this case, no checking is done.

**volume\_sequence\_number (EBCDIC) or file\_section\_number (ASCII)**

A one to four-digit decimal number specifying the volume of a multi-volume file at which you wish to start processing.

If this operand is omitted, the following applies:

- 0001 is used for output files
- no checking is done for input files.

**file\_sequence\_number**

A one to four-digit decimal number specifying the file of a multi-file volume at which you wish to start processing. If the operand is omitted, 0001 is used; no tape repositioning is done for output files.

**generation\_number**

A one to four-digit number specifying the generation number of the file to be processed. If the operand is omitted on output, the system inserts blanks in the appropriate label field. If it is omitted on input, the generation number on the file is not checked by the system.

**version\_number**

A one or two-digit decimal number specifying the version number of the file to be processed. The version number is an extension of the generation number, and the same rules govern its use.

**DISP=NEW | OLD | MOD**

This operand specifies whether a new output file is to be created or an existing file extended. It is meaningful only if:

- The file is to be written by an assembler program assembled under VSE/Advanced Functions Release 1, or a later release, and
- The file is defined in the program using the DTFMT macro with the parameters TYPEFLE=OUTPUT and FILABL=STD.

The specifications have the following meanings:

**NEW**

Specifies that the file is to be created. This is assumed if the DISP operand is omitted.

**OLD**

Specifies that the file already exists and is to be extended. The tape is positioned behind the last record of the existing file. If the file does not already exist, an error situation occurs.

**MOD**

Specifies conditional extension or creation of the file. If the file-id in the



TLBL statement matches the file-id in the HDR1 label on the tape, DISP=OLD is assumed, and the file is **extended**. Otherwise DISP=NEW is assumed, and the file is **created**.

For information on the use of the TLBL statement, see “Job Control for Label Information” in the *z/VSE Guide to System Functions*.

## TPBAL (Telecommunication Balancing)

The TPBAL command allows the operator to change the status of the TP balancing function. Processing may be delayed in the specified number of partitions of the lowest priority.

Without an operand, the TPBAL command displays, on the device assigned to SYSLOG, what partitions are currently being affected by the function.

This command has no effect in a system without page data set, since partition deactivation can only occur at page I/Os for the partition.

Note that Telecommunication Balancing is only in effect for short time intervals, that is, between a TPIN and a TPOUT macro; it is used, for example, by CICS/VSE<sup>®</sup> for ICV processing. It is **not** designed for dynamic (that is, paging-rate dependent) activation and deactivation of low-priority batch partitions.

### AR Format



**n** specifies the number of partitions in which processing can be delayed. The system responds by displaying the partitions that are affected under the new status.

A specification of zero puts the function out of effect.

The maximum number that can be specified must not be greater than the number of static partitions in the system (12) plus the number of classes (maximum 10), minus 1.

After every PLOAD command this value is set to zero, and a new TPBAL command must be issued.

The use of TPBAL is **not** recommended.

## UCS (Load Universal Character-Set Buffer)

The UCS command causes the 240-character universal character set contained in the phase specified by *phasename* to be loaded for the 1403U printer. The 240 EBCDIC characters correspond to the 240 print positions on 1403U trains.

It is the user's responsibility to assemble, link-edit, and catalog his UCS buffer phases, and to mount the new chain or train before the UCS command is executed. The UCS command is not logged on SYSLST.

For further details of phase names, UCB load formats, etc., see the description of the SYSBUFLD program later in this manual.

### JCC Format

```

▶▶—UCS SYSxxx,phasename—┬──┬──┬──┬──▶
                          └──┬──┬──┬──┬──┘
                          ,FOLD ,BLOCK ,NULMSG
  
```

#### SYSxxx

The logical unit assigned to the printer, which must be an IBM 1403 Printer with the UCS feature.

#### *phasename*

The name of the phase containing the 240 EBCDIC characters to be loaded, followed by an 80-character verification message.

#### FOLD

Signifies that the buffer is to be loaded in such a way as to print lowercase bit configurations as uppercase characters.

#### BLOCK

permits any code not represented in the UCS buffer to print as a blank without causing a data check stop.

#### NULMSG

Signifies that the 80-character verification message is not to be printed after the buffer is loaded. If NULMSG is not specified after the UCS buffer has been loaded, the program skips to channel 1, prints 80 characters in the phase specified by the operand *phasename*, and again skips to channel 1. This is to identify the phase, if the *phasename* is incorporated in the verification message. If a chain/train can be identified by a unique character, this character may be included in the verification message to verify that the mounted chain or train is compatible with the UCS buffer contents.

### UNBATCH (Deactivate Foreground Partition)

The UNBATCH command terminates foreground processing and releases the partition (making it inactive). It resets all assignments of the partition to UA, except those for SYSRES, SYSREC and SYSCAT. All temporary and permanent library definitions (LIBDEFs) are dropped. Use the HOLD command if assignments are not to be reset.

#### JCC Format

▶—UNBATCH—▶

The UNBATCH command has no operand.

Following the UNBATCH command, the attention routine accepts BATCH or START commands for the affected partition.

#### Restrictions:

UNBATCH is accepted only:

- From static foreground partitions not controlled by VSE/POWER (not from BG and not from dynamic partitions).
- From SYSLOG - you can gain control of SYSLOG following a PAUSE or STOP command or a // PAUSE statement.
- When no job is active in the partition, that is, after a /& statement has been processed.
- When all tape or disk files assigned to system logical units have been closed.

## UNLOCK (Release Locked Resources)

The UNLOCK command is used to release all resources locked by the specified system. This command should only be used when that system has become inoperative with locks still contained in the lock file. The UNLOCK command is not valid for the system on which it is entered.

### AR Format

►►—UNLOCK SYSTEM=*cpu\_id*—————►►

#### SYSTEM=*cpu-id*

Specifies the CPU-ID of the CPU which has become inoperative. The command will release all locks belonging to the named system. The operator can obtain the CPU-ID from the console printout of the failing system. During system start, IPL message 0I04I identifies the CPU-ID.

The CPU-ID has the format: vvsssssstttt where

vv=X'xx'	Version code for native systems (any two-digit hex number).
vv=X'FF'	Version code for virtual systems running under VM.
ssssss	CPU serial number for native systems, or CPU-ID as specified in the VM/ESA CP command 'SET CPUID', or set by the 'DIRECTORY OPTION' control statement.
tttt	CPU device type or model number of the real machine.

In order to reduce the risk of entering a wrong system-id which would destroy all locks set by the named system, the UNLOCK command causes a verifying message to be displayed on the system console to which the operator has to respond with either YES or NO.

## UPSI (User Program Switch Indicators)

The UPSI statement allows you to set program switches that can be tested by applications during execution.

### JCS Format

▶▶—// UPSI *string*—————▶▶

#### **string**

Is a string of one to eight characters, which correspond to the bit positions of the UPSI byte in the communication region. The specified character string must consist of the characters 0, 1 and X. If you code a 0 in the operand, the corresponding bit in the UPSI byte is set to 0. If you code a 1 in the operand, the corresponding bit in the UPSI byte is set to 1. If you code an X in the operand, the corresponding bit in the UPSI byte remains unchanged. Unspecified rightmost positions in the operand are assumed to be X.

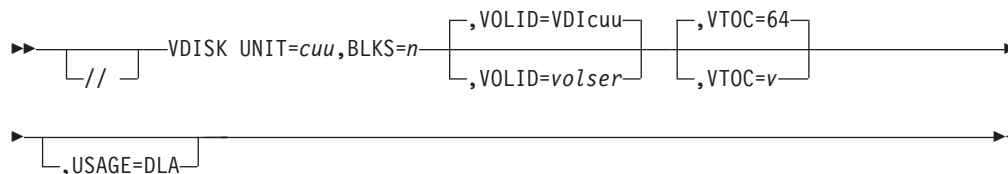
Job control clears the UPSI byte at end of job.

## VDISK (Define Virtual Disk)

The VDISK statement or command can be used to define the layout of a virtual disk and to initialize it implicitly. The virtual disk must have been defined at IPL time with the ADD (FBAV) command and virtual storage must have been allocated to it with the SYSDEF command. VDISK creates a data space for each virtual disk. This data space is cleared to zero and a VTOC is set up indicating an empty disk.

The job control command or statement can be entered in the BG partition only (but at any time).

### JCC, JCS Format



#### UNIT=cuu

Specifies the device number to be used for the virtual disk.

#### BLKS=n

Specifies the size of the virtual disk as a number of 512-byte blocks. *n* can be in the range from 0 to a maximum value of 4,194,240. Since only a multiple of 960 is used as the number of FBA blocks, the specified number is rounded up to a multiple of 960. If, for example, you specify 100, 960 blocks are made available.

Since at most 2GB of storage can be allocated to a data space, the largest multiple of 960 that results in a data space of less than 2GB is 4,194,240. If you do not want to calculate a multiple of 960, just enter a number that suits your needs and get the exact capacity of the virtual disk by using the VOLUME command.

A DVCUP command is implicitly issued after VDISK has been specified with BLKS not equal to zero.

A specification of BLKS=0 indicates that the virtual disk is no longer to be used and the data space is to be deallocated. A value of 0 can be specified only if a device is no longer available for system operation, which requires to use the DVCDN command first. The VOLID and VTOC operands, if specified after BLKS=0 are ignored.

#### VOLID=volser

Specifies the volume serial number of the virtual disk, which can be one to six alphabetic or numeric characters. If fewer than six characters are used, the field is padded on the left with zeros, unless you enclose it in quotes, in which case it is padded on the right with blank characters. (A field enclosed in quotes must not contain blanks or be empty.)

If you do not specify the VOLID operand, the volume serial number is defaulted to VDIcuu.

#### VTOC=v

Specifies the number of 512-byte blocks allocated for the VTOC (which is always put at the end of the virtual disk). For *v*, specify one to three decimal

## VDISK

digits (from 1 to 999). The specified number is rounded to a multiple of eight, because the control-interval size (CISIZE) for the VTOC is 4K.

For each file on the virtual disk and for each additional extent of a file, one label record is written into the VTOC. 28 label records can be written into one control interval of 4K size. Thus, for every 28 label records, eight 512-byte blocks have to be allocated.

If the VTOC operand is omitted, the default VTOC has the following characteristics:

Number of 512-byte blocks: 64, which means  $8 \times 28 = 224$  label records can be written into the VTOC.

Starting location of VTOC: block  $n$  minus 65, where  $n$  is the number of blocks made available in the BLKS operand. If USAGE=DLA has been specified only eight 512-byte blocks (the minimum size) are allocated for the VTOC.

### USAGE=DLA

Specifies that the virtual disk being defined is to hold the label area. The system allocates space for the new (empty) label area starting after the VOL1 label (two 512-byte blocks) of the virtual disk. The VTOC is located at the end of the virtual disk. If the VTOC operand has been omitted, the minimum size (eight 512-byte blocks) is allocated for the VTOC. USAGE=DLA can only be used during ASI (Automated System Initialization) and if no other partitions except BG have been started. It is to be used before any DLBL statements are processed. Up to 2880 blocks can be allocated for the label area. This corresponds to the same capacity as for the label area on a CKD disk. For further capacity considerations refer to "Label Area on Virtual Disk" in the manual *z/VSE Planning*.



## VOLUME (Query Mounted Volumes)

The VOLUME command provides the operator with a short summary of the volumes mounted on disk or tape devices, together with an indication of whether or not a volume is in use, whether a device is shared by another system, and whether it is reserved for VSAM space management usage. The output of the VOLUME command also shows the device capacity and indicates whether the device is a virtual disk.

The record of a tape volume mounted on a tape drive is retained by the system until a new volume is mounted on the drive and this volume is accessed by a program.

### AR Format



**c** Channel address. Information is to be supplied for all disk or tape units on the specified channel.

**cu** Channel and control unit address. Information is to be supplied for all devices on the specified channel and control unit.

#### **cuu**

Device address. Information is to be supplied for the specified device only.

If the operand is omitted, the information is given for all disk or tape volumes at present mounted on the system.

The output of the VOLUME command shows the following (per device):

#### **CUU**

Device address (cuu).

#### **CODE**

VSE device type code and, optionally, the MODE setting, if applicable.

#### **DEVICE-ID**

Device-ID as retrieved from the Sense-ID command, the Read Configuration Data command, or as defined for virtual devices.

#### **VOLID**

Volume serial number of the mounted volume. (DASD or tape.)

#### **USAGE**

USED is displayed, if an assignment for that device exists, or if there is a file on the device. Otherwise UNUSED is displayed. In the case of a CMS minidisk, CMS-D is displayed.

#### **SHARED**

SHARED is displayed, if the device is in use by more than one CPU; otherwise the entry is left blank.

#### **Status-Information**

One of the following:

DOWN  
MOUNT PEND  
READ ONLY

## VOLUME

RESERVED  
NOT READY  
NOT OPER.  
DOWN  
blank if none of the above

### CAPACITY

Device capacity

Figure 13 shows a sample output of the VOLUME command:

```
VOLUME
AR 0015 CUU  CODE DEVICE-ID  VOLID  USAGE  SHARED  STATUS  CAPACITY
AR 0015 100  6E   3390-02  PCT00L  CMS-D      READ ONLY  620  CYL
AR 0015 101  6C   3380-16  PCT192  CMS-D      READ ONLY  80   CYL
AR 0015 190  6C   3380-16  190_09  CMS-D      READ ONLY  120  CYL
AR 0015 191  6C   3380-0A  BAM191  CMS-D      DOWN       3   CYL
AR 0015 19B  6C   3380-0A  $MA19B  CMS-D      READ ONLY  30   CYL
AR 0015 19D  6C   3380-16  CM919D  CMS-D      READ ONLY  200  CYL
AR 0015 19E  6C   3380-16  19E_09  CMS-D      READ ONLY  75   CYL
AR 0015 110  6C   3380-0A  ESA141  USED       40   CYL
AR 0015 111  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 120  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 121  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 130  6C   3380-0A  ESA141  UNUSED     40   CYL
AR 0015 131  6C   3380-0A  HUS131  UNUSED     READ ONLY  30   CYL
AR 0015 132  6C   3380-1E  POWTMP  UNUSED     15   CYL
AR 0015 133  6C   3380-06  POWTMP  UNUSED     15   CYL
AR 0015 134  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 140  6C   3380-16  SYSREC  USED       15   CYL
AR 0015 141  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 142  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 150  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 151  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 152  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 230  6C   3380-0A  PAGEDS  UNUSED     20   CYL
AR 0015 240  6C   3380-06  PAGEDS  USED       60   CYL
AR 0015 241  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 242  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 243  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 244  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 245  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 246  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 247  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 248  6C   3380      UNUSED     DOWN      885  CYL
AR 0015 2F9  90E5  FBAV      UNUSED     DOWN      885  CYL
AR 0015 2FA  90E5  FBAV      UNUSED     DOWN      885  CYL
AR 0015 2FB  90E5  FBAV      UNUSED     DOWN      885  CYL
AR 0015 2FC  90E5  FBAV      UNUSED     DOWN      885  CYL
AR 0015 2FD  90E5  FBAV      UNUSED     DOWN      885  CYL
AR 0015 2FE  90E5  FBAV      UNUSED     DOWN      885  CYL
AR 0015 2FF  90E5  FBAV      UNUSED     DOWN      885  CYL
AR 0015 330  6E   3390-06  BAM330  CMS-D      DOWN       5   CYL
AR 0015 480  54C3  TAPE      UNUSED     NOT OPER.
AR 0015 481  54C3  TAPE      UNUSED     NOT OPER.
AR 0015 482  54C3  TAPE      UNUSED     NOT OPER.
AR 0015 483  54C3  TAPE      UNUSED     NOT OPER.
AR 0015 490  52D3  TAPE      UNUSED     NOT OPER.
AR 0015 491  52D3  TAPE      UNUSED     NOT OPER.
AR 0015 1140I  READY
```

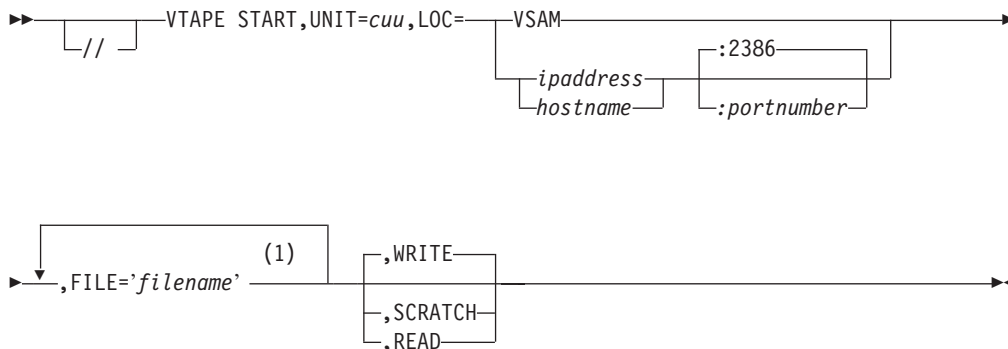
Figure 13. VOLUME Command Output Example

## VTAPE (Define/Release Virtual Tape)

The VTAPE command or statement is used to ask the system to associate a tape unit with a file containing a tape image. Instead reading or writing to a cartridge mounted on a real tape device, the system directs I/O requests to the associated tape image file. This can be either a VSAM file on the VSE system, or a file on another host system running Windows or LINUX.

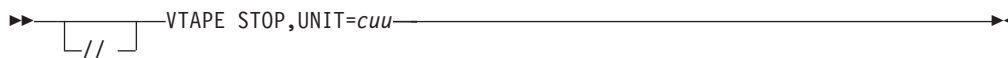
VTAPE can be used to install optional products or apply PTFs from a file containing such a tape image instead from a physical tape.

### JCC, JCS Format



#### Notes:

- 1 FILE='filename' can be specified up to three times.



#### START

Indicates that a tape unit is to be associated with a file containing a tape image. In case of LOC=*ipaddress*, or LOC=*hostname*, a TCP/IP connection to the foreign host is established.

#### STOP

Indicates that an existing association between a tape unit and a tape image file is to be dropped. In case LOC=*ipaddress*, or LOC=*hostname* was specified in the corresponding START command, the TCP/IP connection to the foreign host is closed.

#### UNIT=*cuu*

Specifies the tape unit to be used as virtual tape. *cuu* need not be a physical tape's device number, although it can be. In any case *cuu* must have been ADDED with device type 3480, 3490, or 3490E in the IPL procedure.

#### LOC=VSAM

Indicates that the tape image resides in a VSAM ESDS file on the VSE system. The recommended RECORDSIZE in IDCAMS DEFINE CLUSTER is 16K or larger.

#### LOC=*ipaddress*

Indicates that the tape image resides on a foreign host which is identified by its IP address. *ipaddress* must not be 0.0.0.0. If the contents of *ipaddress* is no valid IP address, the system treats *ipaddress* as hostname.

**LOC=hostname**

Indicates that the tape image resides on a foreign host which is identified by its name. The TCP/IP partition will substitute *hostname* by the associated IP address. *hostname* must not be VSAM and must not contain blanks, commas, colons, or equal signs. Together with the optional *portnumber* the length of the *hostname* must not exceed 100 characters.

**portnumber**

Specifies the TCP/IP port number to be used for the connection. If this operand is omitted, the default port number of 2386 is taken.

**FILE='filename'**

Identifies the file which contains the tape image. **filename** can be up to 100 characters in length, enclosed in quotes.

In case of LOC=VSAM, **filename** is a DLBL name, and corresponding label information must be contained in the system standard label group.

In case of LOC=ipaddress, or LOC=filename, **filename** is the fully qualified file name as appropriate to the file system of the Win/NT or Linux operating system.

Win/NT folder names and file names may contain blanks, therefore **filename** must be enclosed in quotes. A quote within **filename** must be coded as two single quotes, for example

```
FILE='D:\Frank''s\Virtual Tapes\vt021401.001'
```

Win/NT file names can have more than 100 characters in length. Therefore you may specify FILE='filename' twice or even three times. The **filename** information is concatenated in storage, thus allowing for a file name length of 200 or even 300. The following example is equivalent to the one from above:

```
FILE='D:',FILE='\Frank''s\Virtual Tapes\',FILE='vt021401.001'
```

**WRITE**

Specifies, that write access to the file is required. This is the default.

**SCRATCH**

Specifies, that write access to the file is required and that the file is to be written from scratch. An already existing file is overwritten. In case of LOC=VSAM the file must have been defined as reusable.

**READ**

Specifies, that only read access to the file is required. This operand is recommended if for example multiple VSE systems want to simultaneously install the same corrective service residing in a tape file image on one and the same foreign host.

This operand allows to **share** virtual tapes between multiple VSE partitions or multiple VSE systems, which of course is impossible when dealing with physical tapes.

## ZONE (Set Time Zone)

The ZONE statement defines the time difference between local time and Greenwich Mean Time. If no ZONE statement is supplied, job control supplies the zone defined in the IPL SET command.

Locations that are on Greenwich Mean Time need not specify the ZONE statement.

To obtain correct job accounting information, the // ZONE statement should be entered between the /& and the // JOB statement.

### JCS Format

►► // ZONE — EAST/hh/mm —————►  
                   └── WEST/hh/mm ─┘

#### EAST

A geographical position east of Greenwich.

#### WEST

A geographical position west of Greenwich.

#### hh/mm

A decimal value that indicates the difference in hours (00 to 23) and minutes (00 to 59) between local time and Greenwich Mean Time.

## **/(LABEL)**

### **/.(Label Statement)**

The label statement defines a point in the job stream up to which you may want to skip JC statements using a GOTO statement or the GOTO action of an ON statement. When a GOTO is raised, processing continues at the JC statement following the */. label* statement specified.

#### **JCC, JCS Format**

▶▶—/. *label*—————▶▶

Column 1 contains a slash (/) and column 2 a period (.). Column 3 must be blank.

#### **label**

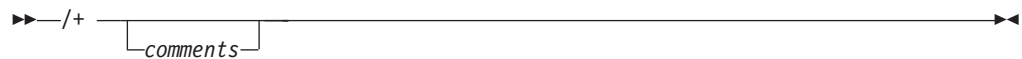
is a name consisting of one to eight alphanumeric characters. The first character must be alphabetic. Symbolic parameters are not allowed in this statement.

The name you specify for label is used as the operand in the corresponding GOTO. The */. label* statement must be coded on the same JC level as the GOTO, that is, both must be within the same procedure, or both outside a procedure (on JC level 0).

**/+ (End-of-Procedure)**

The /+ statement marks the end of a job control procedure. It must be included as the last statement when a procedure is cataloged.

The /+ statement can also be entered on SYSLOG to end the procedure currently running in the appropriate partition.

**JCS Format**


▶▶ /+ \_\_\_\_\_ ▶▶  
           └── comments ─┘

Column 1 contains a slash (/) and column 2 a plus sign (+). Column 3 must be blank.

When used as delimiter on a cataloged procedure, the /+ statement is neither logged nor listed when the procedure is retrieved and included in the job stream. Instead, the following message is written:

EOP procedurename

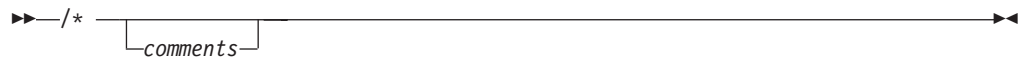
where **procedurename** is the name of the called procedure.

## **/\* (End-of-Data)**

### **/\* (End-of-Data File)**

The end-of-data file statement must be the last statement of each input data file on SYSRDR and SYSIPT. /\* is also recognized for files that Logical IOCS reads from a card reader that is not assigned to SYSIN or SYSIPT.

#### **JCS Format**



Column 1 contains a slash (/) and column 2 an asterisk (\*). Column 3 must be blank.



## /& (End-of-Job)

The /& statement indicates the end of a job. It must be the last statement of each job. Note that /& also forces the end of a procedure (and of any nested procedure).

### JCS Format



Column 1 contains a slash (/) and column 2 an ampersand (&). Column 3 must be blank. If a program attempts to read past the /& on SYSRDR or SYSIPT, an error message is issued. Any comments, beginning in column 36, are printed at end of job. If a job updates a system directory, comments included on the /& statement are not printed.

The end-of-job statement is printed on SYSLST in the following format, where print positions

1-3 contain EOJ;

5-12 the job name;

15-34 the maximum return code if set within this job, otherwise user comments, if any;

35-72 blanks or any user comments; and

69-120 the date, time-of-day, and job duration in the following format:

DATE mm/dd/yyyy, CLOCK hh/mm/ss, DURATION hhhh/mm/ss

**mm/dd/yyyy** can also appear in the order: **dd/mm/yyyy**, if this was specified in the STDOPT command.

However, if a DATE statement with an operand length of 8 has been specified, the EOJ date is shown in the 2-digit year format, for compatibility reasons.

On SYSLOG, the date, time of day, and job duration (the amount of time elapsed between the start and the end of a job) appear in the same format, occupying 52 positions, on the line following the end-of-job statement.

Any temporary control values, such as values of symbolic parameters, ON-conditions, options, assignments and LIBDEFs are reset every time this statement is encountered.

The stop time that the job accounting routines store in the job accounting table is the same as that given for CLOCK at end of job.

End-of-job information is not printed on SYSLST if // OPTION NOLOG has been specified. The NOLOG statement itself is logged on SYSLST.

## \* CP

### \* CP (Submit CP Commands)

The \* CP command allows users with unrestricted command authority to submit CP commands and to receive the related responses. See “Command Authorization in Job Control” on page 54 for detailed information.

#### AR, JCC Format

▶▶—\* CP *string*—————▶▶

#### **string**

is any valid CP command. If specified as job control command, ‘string’ may contain symbolic parameters.

Column 1 contains an asterisk. Column 2 is blank. Column 3 and 4 contain the character string ‘CP’ (Control Program).

Command responses are sent to the origin console.

If the command is part of a VSE/POWER job stream, the command responses are sent to an “ECHO user ID” (provided that such a user ID was specified in the ECHO parameters). If no ECHO user ID was specified, command responses are sent to all master consoles.

## \* (COMMENTS)

The content of the comment statement is printed on SYSLOG. If followed by a PAUSE statement, the statement can be used to request operator action.

### JCS Format



Column 1 contains an asterisk. Column 2 is blank. The remainder of the statement (through column 72) contains any user comments.

**Note:** The :READ statement normally created by VM on top of a job stream (as a result of a PUNCH command) is treated by VSE job control as a comment.

## Job Control Statement Examples

The figures in this section contain examples of job control statement input. In the explanation that follows each example, the numbering of the items corresponds to the numbering at the left of the statements in the example.

Figure 14 to Figure 17 on page 260 show four simple jobs;

Figure 18 on page 261 is an example of conditional job control using the IF statement;

Figure 19 on page 262 shows more complex conditional job control using the IF, ON, and GOTO statements;

Figure 20 on page 263 shows the use of symbolic parameters in procedures;

Figure 21 on page 264 shows the nesting of procedures and the use of parameters in nested procedures.

### General Job Control Examples

The examples on pages 258 to 260 contain four sample jobs. The statements of each job are executed in the sequence as entered. The PHASE, MODE, INCLUDE, and ENTRY statements are linkage editor control statements. These statements are described in detail in Chapter 4, "Linkage Editor." They are included in this discussion to present a more meaningful example.

```

1 // JOB U81SDC                               UNLOAD SEQUENTIAL DISK TO TAPE
2 // ASSGN SYS004,111                          INPUT MASTER FILE
  // ASSGN SYS005,112                          (2 EXTENTS)
  // ASSGN SYS006,380,C8                       BACKUP TAPE DUAL DENSITY 9-TRK
3 // DLBL SDUNLD,'SEQUENTIAL FILE',1999/206,SD
  // EXTENT SYS004,338002,1,0,1900,380
  // EXTENT SYS005,338003,1,1,76,570
4 // EXEC SD008,REAL,SIZE=60K                 RUN IN REAL USING 60K
5 // MTC RUN,SYS006
6 * OPERATOR - TAPE ON 380 - LABEL, REMOVE RING AND ARCHIVE
7 /&

```

Figure 14. General Job Control Examples Part 1

1. JOB statement.
2. ASSGN statements for disks and tape.
3. DLBL and EXTENT statements to define a sequential disk file with two extents on separate volumes.
4. EXEC statement for a program in a sublibrary that is to be executed in real mode, using 60K of processor storage allocated to BG.
5. MTC command to rewind and unload the tape just created.
6. Message to notify operator that tape handling is required.
7. End-of-job indicator.

```

1 // JOB R61ASSM          OBJECT DECK TO TAPE - CATALOG
  *                      IN SUBLIBRARY
  * CREATE A MAP OF STORAGE ON SYSLOG
2 MAP
3 ASSGN SYS012,UA        CLEAR PREVIOUS TAPE ASSIGN
  // ASSGN SYSPCH,381    ASSIGN SYSPCH TO TAPE
4 // OPTION DECK,LIST,XREF,NOEDECK
5 CATALR MOD207
6 // EXEC ASMA90
7 ...                   ASSEMBLER SOURCE HERE
  /*
8 // MTC WTM,SYSPCH,02   WRITE TAPEMARK AND
  // MTC REW,SYSPCH      REWIND SYSPCH TAPE
9 // ASSGN SYSIPT,381    ASSEMBLER OUTPUT TO LIBR INPUT
10 // EXEC LIBR,PARM='ACCESS S=LIB1.S2'  SUBLIB FOR OBJ
  /*                     CATALOG FROM SYSIPT
  // MTC RUN,381         REWIND/UNLOAD SYSIPT
11 /&                   EOJ R61ASSM

```

Figure 15. General Job Control Examples Part 2

1. JOB statement.
2. MAP command to print a map of storage allocations on SYSLOG.
3. ASSGN statements to release previous tape assignment, and temporarily assign SYSPCH to that tape.
4. OPTION statement to specify options that are different from the permanent options.
5. Statement that will be transferred by job control to the SYSPCH file (tape on 381). This tape can then, after creation of the object deck, be used as input to the program to catalog it as a library member of the type OBJ.
6. EXEC statement for the system assembler.
7. Source deck as input to the system assembler and /\* (end-of-data).
8. MTC statements to write a tapemark and rewind the tape on 381. This tape is now positioned for later use as SYSIPT.
9. The tape on 381 is temporarily re-assigned as SYSIPT for the librarian catalog run.
10. The librarian program is called. The sublibrary in which the object module is to be cataloged is specified in the ACCESS command passed in the PARM operand. The CATALOG statement is read from SYSIPT with the assembler output.
11. End-of-job indicator with a comment. SYSIPT returns to its permanent assignment.

## General Examples

```
1 // JOB K13CATL                LINK MODULES INTO A
*                               SUBLIBRARY
2 LIBDEF PHASE,SEARCH=(LIB1.S2,LIB1.S3),CATALOG=LIB1.S3,PERM
3 // OPTION CATAL
4 PHASE PROGX03,*
4 MODE AMODE(ANY)
4 INCLUDE MOD207
  INCLUDE
  ...                               OBJECT DECK INCLUDED HERE
5 /*
6 ENTRY MD207B
7 // EXEC LNKEDT
8 /&
```

Figure 16. General Job Control Examples Part 3

1. JOB statement.
2. Permanent definition of sublibrary chains from which programs are to be loaded, and into which phases are to be cataloged.
3. OPTION statement to specify that the phase produced by the linkage editor is to be cataloged.
4. PHASE, MODE, and INCLUDE statements are input to the linkage editor. The first INCLUDE statement calls the module previously cataloged in the sublibrary and the second (with a blank operand) is followed by an object deck to be included.
5. /\* indicates the end of the object deck, not the end of input to the linkage editor.
6. ENTRY statement input to the linkage editor specifying an entry point for the PHASE PROGX03.
7. EXEC statement for the linkage editor.
8. End-of-job indicator.

```
1 // JOB E40
2 // ASSGN SYSLST,PRINTER        ASSIGN SYSLST TO ANY PRINTER
3 // ASSGN SYS004,(380,381,382)  ASSIGN TO TAPE WITHIN THIS RANGE
  // ASSGN SYS006,(280:282)
4 // ASSGN SYS005,SYS004        ASSIGN SYS005 as SYS004
  // EXEC MYPROG
5 /&
```

Figure 17. General Job Control Examples Part 4

1. JOB statement.
2. ASSGN statement for SYSLST, which may be any printer.
3. SYS004 should be assigned to a tape on 380, or 381 (if 380 is not available), or 382 (if both 380 and 381 are not available). Similarly, for SYS006 and 280, 281, 282.
4. Assign SYS005 to the same unit as SYS004 (described in 3).
5. End-of-job indicator.

## Conditional Job Control: Example of IF Statement

```
1 // JOB A3243           EXAMPLE OF IF STATEMENT
2 // EXEC PGM1          FIRST PROGRAM
3 IF $RC=0 THEN        TEST RETURN CODE OF PGM1
4 // EXEC PGM2          IF RC OF PGM1 WAS 0, RUN PGM2
5 /&
```

*Figure 18. The Use of the IF Statement*

Explanation of the sequence numbers in Figure 18:

1. JOB statement for Job A3243.
2. If program PGM1 is not canceled and does not terminate abnormally, it will set a return code from 0 to 4095. This return code is used to control the processing of the following JCL statements.
3. The return code of the preceding step is tested, and the next statement is executed only if the condition is true. Otherwise, the statement is skipped.
4. Program PGM2 is executed only if the condition of the preceding IF statement was true.
5. End-of-Job indicator. All conditional JCL information is reset to default values.

## Conditional Job Control: Example of ON, IF and GOTO Statements for Abnormal Termination

```
1 // JOB A3244          EXAMPLE OF ON, IF AND GOTO STATEMENTS
2 ON $ABEND GOTO AB    FOR ABNORMAL TERMINATION
  // EXEC PGM1
3 IF $RC > 4 THEN     TEST RETURN CODE OF PGM1
  // EXEC PGM2       IF RC OF PGM1 WAS GREATER THAN 4
4 GOTO $EOJ          SKIP ABTERM STEP - END JOB
5 /. AB             SKIP TO HERE IN CASE OF ABNORMAL TERM.
6 // EXEC ABTERM    ONLY IN CASE OF ABNORMAL TERMINATION
/&
```

*Figure 19. IF, ON and GOTO Statements for Abnormal Termination*

Explanation of the sequence numbers in Figure 19:

1. JOB statement for Job A3244.
2. If any of the following steps terminate abnormally, job control skips all statements up to the label AB. This statement overrides the default condition ON \$ABEND GOTO \$EOJ.
3. If the return code of PGM1 is greater than 4, the next statement is executed.
4. If this statement is processed, the rest of the statements in the job are skipped. This would be the case if neither PGM1 or PGM2 terminated abnormally.
5. If the condition of the ON statement occurred (one of the steps terminated abnormally), processing continues at this point.
6. This program will be executed only if an abnormal termination occurs.



## Parameterized Procedure Example

Job stream as submitted to job control:

```

1 // JOB A3245                               EXAMPLE OF PARAMETERIZED PROCEDURE
2 // SETPARM RC1=                             DEFINE AND NULLIFY PARAMETER
3 // EXEC PROC=UPDAT,RC1,SER=338006          CALL PROC, PASS PARAMETERS
9 // EXEC PGM=EVALUATE,PARM='&RC1'         CALL PROGRAM, PASS PARAMETER
10 /&

```

Procedure UPDAT as cataloged:

```

4 // PROC DEV=3380,SER=338006             DEFINE AND INITIALIZE PARAMETERS
5 // ASSGN SYS008,&DEV,TEMP,VOL=&SER,SHR
6 // EXEC PGM=UPDATE                       CALL PROGRAM
7 SETPARM RC1=$RC                         SAVE RC OF PGM UPDATE
8 /+

```

Note: The index numbering on the left is in processing sequence.

Resulting flow of execution:

```

1 // JOB A3245                               EXAMPLE OF PARAMETERIZED PROCEDURE
2 // SETPARM RC1=                             DEFINE AND NULLIFY PARAMETER
3 // EXEC PROC=UPDAT,RC1,SER=338006          CALL PROC, PASS PARAMETERS
4 // PROC DEV=3380,SER=338006             DEFINE AND INITIALIZE PARAMETERS
5 // ASSGN SYS008,&DEV,TEMP,VOL=&SER,SHR
6 // EXEC PGM=UPDATE                       CALL PROGRAM
7 SETPARM RC1=$RC                         SAVE RC OF PGM UPDATE
8 /+
9 // EXEC PGM=EVALUATE,PARM='&RC1'         CALL PROGRAM, PASS PARAMETER
10 /&

```

Figure 20. The Use of a Parameterized Procedure

Explanation of the sequence numbers in Figure 20:

1. JOB statement for Job A3245.
2. Definition of parameter RC1, and assignment of null string.
3. Procedure UPDAT is called. The parameter SER is defined for this call of the procedure, and passed to the procedure. The parameter SER is defined for this procedure, and the value '337506' is assigned to it. The existing parameter RC1 is passed to the procedure.
4. Default values for the procedure UPDAT are defined. The parameter DEV is not specified in the procedure call (3), so the default value '3375' is used. The default value of SER is not used, because it has been specified in the procedure call with the value '337506'.
5. The device type and volume serial number in the ASSGN statement are specified as symbolic parameters. Job control substitutes the values '3375' and '337506' for DEF and SER, respectively.
6. Execution of the program UPDATE. This step terminates with a return code.
7. The return code of the program UPDATE is assigned to the parameter RC1. This parameter will still be available after the procedure UPDAT has been terminated, as it was passed to UPDAT in the calling EXEC statement.
8. End of the procedure UPDAT.
9. Execution of the program EVALUATE. The symbolic parameter RC1 is used as a program parameter, thus making the return code of the program UPDATE in the procedure UPDAT available to the program EVALUATE.
10. End-of-Job indicator.

### Parameterized Procedures and Procedure Nesting Example

Job stream as submitted to job control:

```
1 // JOB A3246           EXAMPLE OF NESTED PROCEDURES
2 // EXEC PROC=PROC1    CALL PROCEDURE PROC1
11 // EXEC PGM=LISTPGM  CALL PROGRAM LISTPGM
12 /&                   END OF JOB A3246
```

Procedure PROC1 as cataloged:

```
3 // EXEC X240          CALL PROGRAM X240
4 // EXEC PROC=X24,NO=2 CALL PROC X24, PASS PARAMETER
9 // EXEC X243          CALL PROGRAM X243
10 /+                   END OF PROCEDURE PROC1
```

Procedure X24 as cataloged:

```
5 // PROC NO=1         INITIALIZE PARAMETER NO TO 1
6 // EXEC X241         CALL PROGRAM X241
7 IF NO=2 THEN        TEST PARAMETER NO
  // EXEC X242         CALL PROGRAM X242 ONLY IF NO=2
8 /+                   END OF PROCEDURE X24
```

Note: The index numbering on the left is in processing sequence.

Resulting flow of execution:

```
1 // JOB A3246           EXAMPLE OF NESTED PROCEDURES
2 // EXEC PROC=PROC1    CALL PROCEDURE PROC1

3 // EXEC X240          CALL PROGRAM X240
4 // EXEC PROC=X24,NO=2 CALL PROC X24, PASS PARAMETER

5 // PROC NO=1         INITIALIZE PARAMETER NO TO 1
6 // EXEC X241         CALL PROGRAM X241
7 IF NO=2 THEN        TEST PARAMETER NO
  // EXEC X242         CALL PROGRAM X242 ONLY IF NO=2
8 /+                   END OF PROCEDURE X24

9 // EXEC X243          CALL PROGRAM X243
10 /+                   END OF PROCEDURE PROC1

11 // EXEC PGM=LISTPGM  CALL PROGRAM LISTPGM
12 /&                   END OF JOB A3246
```

*Figure 21. Use of Nested Procedures*

Explanation of the sequence numbers in Figure 21:

1. JOB statement for Job A3246.
2. Procedure PROC1 is called.
3. Execution of program X240.
4. Procedure X24 is called. The parameter NO is defined, and the value '2' is assigned to it. (Calling a procedure from another procedure is called "procedure nesting".)
5. The default value of NO=1 is not used because this parameter has been defined in the procedure call.
6. Execution of program X241.
7. The IF statement examines the present value of the parameter NO. If the condition were not true, the next statement would be skipped. In this case, the condition is true, and the next statement is executed.

8. End of procedure X24. Control returns to procedure PROC1, at a point immediately after the EXEC PROC statement (4) which caused control to be passed to procedure X24.
9. Execution of program X243.
10. End of procedure PROC1. Control returns to SYSRDR at a point immediately after the EXEC PROC statement (2) which called PROC1.
11. Execution of program LISTPGM.
12. End-of-Job indicator.

## Nesting Example

---

## Chapter 4. Linkage Editor

The linkage editor prepares programs for execution and accepts as input the relocatable object modules produced by the language translators and object modules produced by the librarian PUNCH command. It processes these modules into program phases, which may be executed immediately or cataloged into a library.

If object modules from a sublibrary are to be link-edited, the sublibrary must be defined with the statement:

```
// LIBDEF OBJ,SEARCH=lib.sublib
```

The sublibrary into which phases are to be cataloged must be defined with the statement:

```
// LIBDEF PHASE,CATALOG=lib.sublib
```

If the correct search chain and catalog sublibrary definitions have been made permanent for the partition, you need not include them in the linkage editor job.

Now you can define the linkage editor input and then call the linkage editor program with the following job control statement:

```
▶▶—// EXEC LNKEDT—▶▶
```

```
  ,PARM=' MSHP  
          AMODE=24  
          AMODE=31  
          AMODE=ANY  
          RMODE=24  
          RMODE=ANY
```

If the present run of the linkage editor will replace an existing phase that is under control of the Maintain System History Program (MSHP), the EXEC statement must contain the parameter MSHP.

The AMODE/RMODE values specify the addressing mode and residence mode for all phases linked in the link-edit job. AMODE determines the addressing mode for the phase entry points and RMODE determines where the phases can reside in virtual storage.

The AMODE/RMODE parameters override the mode information derived from the ESD data for the control section. Note that if a MODE statement is specified for a phase, this specification is decisive, overriding ESD and PARM field definitions for this phase.

If the AMODE or RMODE parameter occurs more than once in the PARM field, the last valid parameter is used.

If only one value, either AMODE or RMODE, is specified in the PARM field, the other value is implied according to the following table:

## Linkage Editor

Specified Value	Implied Value
AMODE=24	RMODE=24
AMODE=31	RMODE=24
AMODE=ANY	RMODE=24
RMODE=24	see below
RMODE=ANY	AMODE=31

If only RMODE=24 is specified in the PARM field, no overriding AMODE is implied; instead, the AMODE value in the ESD data for the entry point is used.

The following combinations of AMODE and RMODE are possible in the PARM field of the EXEC statement:

	RMODE=24	RMODE=ANY
AMODE=24	Valid	Invalid
AMODE=31	Valid	Valid
AMODE=ANY	Valid	Invalid

If the AMODE/RMODE combination is invalid, the linkage editor issues a warning message on SYSLST and ignores the AMODE/RMODE parameters.

The actual AMODE/RMODE of a phase is shown in the linkage editor map. An example is shown under "ACTION: Print Linkage Editor Map" in the *z/VSE Diagnosis Tools*.

---

## YEAR 2000 Support

The linkage editor extends the year representation to 4 digits (retrieved from field JOBDATWC in partition COMREG). Programs scanning the printout of the linkage editor may have to be adapted because of the new date format.

There is one exception to this rule. The linkage editor will not use 4 digits to display the year if the // DATE job control statement has been used, specifying only 2 digits for the year. In this case, the header lines also show a 2-digit year (from field JOBDAT in COMREG, for compatibility reasons).

The following changes apply:

1. One header line is printed on each new page on SYSLST when reading input from SYSLNK and printing input statements on SYSLST. This header line contains the date with a 4-digit year, instead of 2 digits followed by 2 blanks. The rest of the header line and all the following lines remain unchanged.
2. One header line is printed on each new page on SYSLST when printing the linkage editor map, displaying phase names, CSECT names, linked module names and so on. In this header line, the date is contained in the very first 8 bytes (digits and slashes), followed by just one blank before the next header text (PHASE) begins.

If 4 digits are displayed for the year, the date is displayed using 10 bytes instead of 8 bytes. All header information following the date is moved 2 bytes to the right.

The header text "PHASE" following the date is a column header, which means that the information in the subsequent lines is adjusted under it. So if 4 digits are used for the year, all lines following the header line are also moved 2 bytes to the right.

---

## Linkage Editor Return Codes

The Linkage Editor passes the following return codes to job control:

```
0 = successful
2 = warning message issued but phases are cataloged
4 = warning or error message issued but phases are cataloged
8 = single phase not replaced
16 = severe error(s), phases are not cataloged
```

**Note:** If the linked phase contains only unresolved address constants for external symbols identified by the WXTRN assembler instruction, the Linkage Editor returns (starting with VSE/ESA 1.3.0) return code 2 (instead of return code 4 as in previous releases).

---

## Language Translator Modules

The input to the linkage editor consists of linkage editor control statements and object modules. Each module is the output of a complete language translator run. It consists of dictionary and text records for one or more control sections.

The dictionaries contain the information necessary for the linkage editor to resolve references between different modules and to perform program relocation. The text consists of the actual instructions and data fields of the module.

Six statement types can be produced, by the language translators or by the programmer, to form a module. They appear in the following order:

Stmt Type	Contents/Purpose
ESD	External symbol dictionary
SYM	Ignored by linkage editor
TXT	Text
RLD	Relocation list dictionary
REP	Replacement text supplied by the programmer if necessary.
END	End of module.

For the format of each of these statements (except SYM), see the Appendix.

The **external symbol dictionary** contains control section definitions and inter-module references. When the linkage editor has the ESDs from all modules, it can relocate the sections and resolve the references. For the entries contained in the external symbol dictionary, see "External Symbol Dictionary" on page 450.

The **relocation list dictionary** identifies portions of text that must be modified on relocation (address constants). Unresolved address constants are set to zero in relocatable phases.

---

### Linkage Editor Control Statements

In addition to the language translator output previously listed, input for the linkage editor includes linkage editor control statements. These statements are briefly discussed below and described in detail further on in this section.

#### **ACTION**

Specifies options to be taken.

#### **ENTRY**

Provides an optional transfer address for the first phase and ends the linkage editor input.

#### **INCLUDE**

Signals that an object module or a number of CSECTs contained in an object module are to be included in the phase currently being processed.

#### **MODE**

Assigns the addressing mode (AMODE) for the entry point of a phase and the residence mode (RMODE) for a phase.

#### **PHASE**

Indicates the beginning of a phase. It gives the name of the phase and the storage address where it is to be loaded.

---

### General Control Statement Format

The linkage editor control statements must be coded in the following format:

- The operation field must be preceded by one or more blanks.
- The operation field must be separated from the operand field by at least one blank position.
- The operand field is ended by the first blank position. It cannot extend past position 71.

---

### Control Statement Placement

The following describes the placement of control statements and object code in the job stream and in library members. See "Preparing Input for the Linkage Editor" in the manual *z/VSE Guide to System Functions* for a general description of preparing input for the linkage editor.

ACTION statements can be placed at any position in the input stream. The options they define are effective either for the entire job or dependent on the placement of the statement.

PHASE statements determine the beginning of a phase, i.e. modules and control sections following the statement are accumulated in this phase. The end is determined by another PHASE statement or the ENTRY statement.

MODE statements define phase characteristics. They must follow the PHASE statement they belong to.

INCLUDE statements can be placed at any position in the job stream or in library members. They determine which object code is to be included in a phase, and they can switch the source of input records from SYSLNK to a library member or from one library member to another.

ENTRY statements determine the end of Linkage Editor input, i.e. ENTRY is the last control statement before EXEC LNKEDT. Job control adds if it has been omitted.



In the following examples SYSRDR and SYSIPT are assumed to be assigned to the same unit. Object code (OC) is assembler or compiler output consisting of ESD, TXT, and END records.

### Example 1

```
// OPTION CATAL           | read from SYSRDR
INCLUDE                   |
  ACTION NOAUTO           |
  PHASE A,*               |
  MODE AMODE(ANY)         |
  OC1                     |
  OC2                     |
  PHASE B,*               | read from SYSIPT
  OC3                     |
  OC4                     |
  OC5                     |
  /*                       |
INCLUDE M                 |
PHASE C,*                 |
MODE RMODE(24)           |
INCLUDE N                 | read from SYSRDR
INCLUDE X
INCLUDE S
ENTRY
// EXEC LNKEDT           |
```

Contents of X:

```
PHASE D,*
ACTION SMAP
INCLUDE Y
INCLUDE R
END (hex'02C5D5C4..')
```

Contents of Y:

```
INCLUDE P
INCLUDE Q
OC6
```

M, N, P, Q, R, and S are object modules cataloged in the library. The Linkage Editor will produce following phases:

- Phase A consisting of OC1 and OC2;
- Phase B consisting of OC3, OC4, OC5, and M;
- Phase C consisting of N;
- Phase D consisting of P, Q, OC6, R, and S.

### Example 2

```
// OPTION CATAL
PHASE A,*
INCLUDE ,(CS8,CS2,CS5)
INCLUDE
  ESD ..CS1..CS2..CS4..CS8..CS9..
  TXT
  ...
  END
  /*
ENTRY
// EXEC LNKEDT
```

The resulting phase A will consist of control sections CS2 and CS8 (in this sequence).

### Example 3

```
// OPTION CATAL
  PHASE A,*
  INCLUDE ,(CS8)
  INCLUDE ,(CS2)
  INCLUDE ,(CS5)
  INCLUDE
  ESD ..CS1..CS2..CS4..CS8..CS9..
  TXT
  ...
  END
  /*
  ENTRY
// EXEC LNKEDT
```

The resulting phase A will consist of control sections CS8 and CS2 (in this sequence).

### Example 4

```
// OPTION CATAL
  INCLUDE X
// EXEC LNKEDT
```

Contents of X:

```
  PHASE A,*
  INCLUDE M,(CS3,CS2)
  INCLUDE N,(CS7)
  INCLUDE N,(CS5)
  INCLUDE Y,(CS9)
  INCLUDE Q
  END (hex'02C5D5C4..')
```

Contents of Y:

```
  INCLUDE P
  END (hex'02C5D5C4..')
```

The resulting phase A will consist of control sections CS2 and CS3 from module M, CS7 and CS5 from module N, CS9 from module P, and module Q.

## ACTION

The ACTION statement is used to indicate linkage editor options; it defines certain conditions for the remaining job step. ACTION statements can be placed at any position in the job. If multiple operands are required, they may either be specified in one statement, separated by commas, or be placed separately in several statements.



### Notes:

- 1 Operands must be specified in capital letters.  
At least one blank must precede ACTION.

### MAP

Requests the linkage editor to write to SYSLST a map of virtual storage, which can be used for problem determination. The map contains the name of every CSECT within each phase and the name of every entry within each CSECT. For an example of a partition storage map, together with a description of how to interpret it, see "ACTION: Print Linkage Editor Map" in the *z/VSE Diagnosis Tools*.

The MAP (or NOMAP) option becomes valid immediately; thus the linkage editor map can be structured into parts that are desired for printout and those that are to be suppressed.

If the MAP operand is specified, SYSLST must be assigned. If the ACTION statement is not used, MAP is assumed when SYSLST is assigned, and NOMAP is assumed when SYSLST is not assigned.

### NOMAP

Indicates that the MAP option should not take effect. The system lists all linkage editor error diagnostics on SYSLOG.

### NOAUTO

Indicates that the AUTOLINK function is to be suppressed for the current and all subsequent phases of the job step. If specified prior to the first PHASE statement, AUTOLINK is suppressed for all phases in the job step.

The NOAUTO operand in a PHASE statement indicates to the linkage editor that AUTOLINK is to be suppressed for that phase only. If an entire program requires NOAUTO, then specifying ACTION NOAUTO cancels AUTOLINK during link editing of the entire program, thereby eliminating the necessity of specifying NOAUTO in each PHASE statement.

**Note:** When a weak external reference (WX) is encountered, it is treated in the same manner as a normal external reference with NOAUTO.

## **ACTION**

### **CANCEL**

Cancels the job automatically if any of the messages 2100I through 2179I (except 2139I) occur. These are errors causing RC=2 or RC=4. More severe errors (messages 2180I through 2198I) cause abnormal termination with RC=16 and are not influenced by ACTION CANCEL. If CANCEL is not specified, the job continues. The CANCEL option becomes effective independent of the position of the ACTION statement.

### **SMAP**

Indicates that, in addition to the standard virtual storage map in which the control sections are ordered by load address, a listing of the CSECT names ordered alphabetically is also generated. This list may be useful if you want to locate a CSECT by its name and the phase consists of many CSECTs. The SMAP option becomes effective independent of the position of the ACTION statement.

### **ERRLMT(nnnn)**

To prevent indefinite error loops, the number of error messages is limited to a predefined value of 256, which is also the default value. You can override this limitation by specifying the value nnnn, which can be any decimal number from 1 through 9999. When this limit is exceeded, an information message is issued and the linkage editor job step is terminated with return code 16.

## ENTRY

Every program, as input for the linkage editor, is terminated by an ENTRY statement. Job control writes an ENTRY statement with a blank operand on SYSLNK when EXEC LNKEDT is read; this causes the load address of the first phase to be used as the transfer address if no transfer address is specified on the END cards of the input OBJ modules. (If a transfer address is specified on an END card of the OBJ modules contributing to the first phase, the transfer address on the **first** END card containing that address is used as transfer address). It is necessary to supply the ENTRY statement only if you specifically request another entry point.



### Notes:

- 1 Options must be specified in capital letters.  
At least one blank must precede ENTRY.

### entrypoint

Specifies the name (label) of an entry point. It must be the name of a CSECT or a label definition (source ENTRY) defined in the first phase. This address is used as the transfer address to the first phase in the program. If the operand field is blank, the linkage editor uses as a transfer address the first significant address provided in an END record encountered during the generation of the first phase. If no such operand is found on the END card, the transfer address is the load address of the first phase.



**Restrictions:**

- Unnamed CSECTs cannot be selected in a namelist.
- The number of names in a namelist is limited to five.
- The object code for a submodular INCLUDE without modulename must follow in the same nesting level.
- Submodular INCLUDEs with modulename cannot be nested.
- Between an "INCLUDE ,(namelist)" statement and the object code only further INCLUDE statements of the same type are allowed. Other control statements may lead to unexpected results.

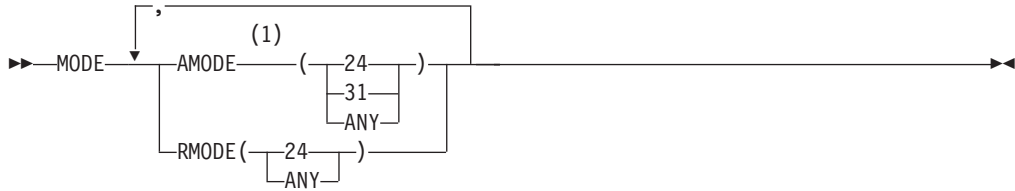
## MODE

The MODE statement is used to assign the addressing mode (AMODE) for the entry point of a phase and the residence mode (RMODE) for a phase, which indicates where the phase can reside in virtual storage.

The mode value overrides any AMODE|RMODE value specified as an operand in the PARM field of the EXEC LNKEDT statement. The RMODE value assigned by the MODE statement also overrides the RMODE accumulated from the input control sections and private code. The AMODE value assigned by the MODE statement also overrides the addressing mode found in the ESD data for the control section or private code within which the entry point is located.

The MODE statement must follow the PHASE statement of a phase.

If more than one MODE statement is encountered in the link-edit for a phase, the mode specification from the first valid MODE statement is used.



**Notes:**

- Options must be specified in capital letters.  
At least one blank must precede MODE.

**mode**

Specifies the addressing mode and/or the residence mode of a phase. If a mode specification occurs more than once within the same MODE statement, the last valid mode specification is used.

If only one value, either AMODE or RMODE, is specified in the MODE statement, the other value is implied according to the following table:

Specified Value	Implied Value
AMODE(24)	RMODE(24)
AMODE(31)	RMODE(24)
AMODE(ANY)	RMODE(24)
RMODE(24)	see below
RMODE(ANY)	AMODE(31)

If only RMODE(24) is specified in the MODE statement, no overriding AMODE is implied; instead, the AMODE value in the ESD data for the entry point is used.



The following combinations of AMODE and RMODE are possible on the MODE control statement:

	RMODE(24)	RMODE(ANY)
AMODE(24)	Valid	Invalid
AMODE(31)	Valid	Valid
AMODE(ANY)	Valid	Invalid

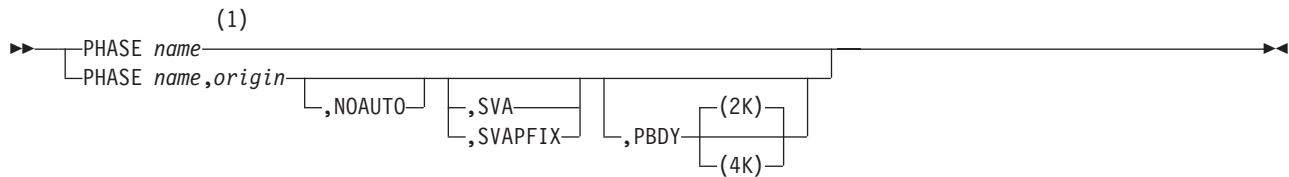
If the AMODE/RMODE combination is invalid, the linkage editor issues a warning message on SYSLST and ignores the MODE control statement.

## PHASE

### PHASE

This statement provides the linkage editor with a phase name and an origin point for the phase.

The phase name is used to catalog the phase into a sublibrary. It is also used when the phase is retrieved for execution. The PHASE statement must precede the first object module of each phase processed by the linkage editor. Any object module not preceded by a PHASE statement is included, together with the preceding object modules, in the current phase. If no PHASE statement is used, or if the first PHASE statement is in error, the linkage editor generates a dummy statement. This allows testing of the program when the LINK option is used. However, the program with the dummy PHASE statement cannot be cataloged in a sublibrary; when the CATAL option is used, the job is terminated with return code 16.



#### Notes:

- Options must be specified in capital letters

At least one blank must precede PHASE. The operands have the following meaning:

#### name

Specifies the name of the phase. One to eight alphanumeric (0-9, A-Z, #, \$, and @) characters are used as the phase name. For phases which are to be cataloged, use the librarian conventions for member names. The name may not be ALL, S, or ROOT.

Each single-phase program should be unique in the first four characters of its phase name, because all phases whose names start with the same four characters will be treated as a multiphase program. When a multiphase program, is being fetched, the partition must be large enough to contain the largest phase, unless you specify SIZE=name in the EXEC statement.

#### origin

Specifies the load address of the phase. The load address can be in one of the following forms:

- symbol[(phase)][+relocation]
- \*[+relocation]
- S[+relocation]
- ROOT
- +displacement

Items 1 to 4 specify a relative address, item 5 an absolute address.

A phase can be made relocatable if its origin is specified as a relative address (formats 1-4 above). However, if the address is relative to another phase which is not relocatable, the new phase will not be relocatable. If origin is not specified, the phase is made relocatable.

The elements that make up the various forms that specify the origin are the following:

1. **symbol:** May be a previously defined phase name, control section name, or external label (the operand of an ENTRY source statement).

**(phase):** If **symbol** is a previously defined control section name or a previously defined external label that appears in more than one phase, **phase** (in parentheses) directs the linkage editor to the phase that contains the origin. The phase name must have been defined previously.

**relocation:** Indicates that the origin of the phase currently being processed will be set relative to the symbol by a relocation term consisting of:

- a + or a - immediately followed by: X'hhhhh' (one to six hexadecimal digits);
- a + or a - immediately followed by: ddddddd (one to eight decimal digits);
- a + or a - immediately followed by: nK, (where n is the number of kilobytes).

2. \*: Indicates that, for the first PHASE statement processed, the origin is to be the first doubleword storage address after the partition save area, or the area assigned to the COMMON pool (if any).

For successive PHASE statements, the linkage editor assigns the storage location immediately following the end of the preceding phase (with forced doubleword alignment) as an origin for the phase.

**relocation:** Indicates relocation of the phase relative to the next storage location of the virtual partition. The format is as specified in item 1.

3. **S:** if **S** is specified, the origin is determined in the same manner as for the first PHASE statement in item 2.

**relocation:** Indicates relocation of the phase relative to the start of the virtual partition as described in item 2.

4. **ROOT:** Tells the linkage editor that the phase that follows is a root phase. The storage address assigned to the root phase is determined in the same manner as the first PHASE statement in item 2. Only the first PHASE statement in the linkage editor input can specify ROOT. If a control section (CSECT) appears in the root phase, other occurrences of the same control section are ignored and all references are resolved to the control section in the root. Control sections are not duplicated within the same phase. If any subsequent phase overlays any part of the ROOT phase, a warning diagnostic is displayed on SYSLST if ACTION MAP is in effect. Refer also to the description of the ACTION statement earlier in this section.

5. **+displacement:** Allows the origin (loading address) to be set at a specified location. The origin point is an absolute address, relative to zero.

**+displacement** must be:

+X'hhhhh' (one to six hexadecimal digits),  
+ddddddd (one to eight decimal digits), or  
+nK (where n is the number of kilobytes).

A displacement of zero (+0) denotes a self-relocating program.

To link-edit a program for execution in real mode, you can

- Link-edit the program so that it can be relocated to a real partition when it is loaded.
- Write the program to be self-relocating.

## PHASE

- Link-edit the program with a PHASE statement that contains the absolute address of the location within the real partition where the program is to be loaded.

If a COMMON area (such as in FORTRAN programs) is used, the length of the largest COMMON is added to every phase origin, even if the origin is given as an absolute value. COMMON is located at the beginning of the phase with the lowest origin address (if multiple phases).

**Note:** If the origin address supplied is not on a doubleword boundary, the linkage editor automatically increments that address to the next doubleword boundary.

### NOAUTO

Indicates that the Automatic Library Lookup (AUTOLINK) feature is suppressed for the current phase. If you want to suppress it for the remaining or for the entire program, specify ACTION NOAUTO.

### SVA

Indicates that the phase is SVA-eligible. This means that the phase must be re-entertainable and relocatable. When this phase is cataloged into sublibrary IJSYSRS.SYSLIB, the linkage editor will also have the phase loaded into the SVA if the phasename was listed in the SDL with the SVA option. If the linkage editor finds that a phase that is specified with the SVA option is not relocatable, an error message is issued and the SVA option is ignored.

### SVAPFIX

Indicates that the phase is SVA-eligible and that it is to be loaded into PFIxed storage.

The use of this operand should be very carefully evaluated since the corresponding real storage is removed from the page pool for all VSE users/partitions.

### PBDY[(2K|4K)]

Indicates that the referenced phase is to be linked to start on a (2K- or 4K-) page boundary. The K-specification may be of significance if your program uses the overlay technique and requires the overlay phases to start on a 4K-page boundary. The 2K- or 4K-specification is not recommended for the first (root) or only phase of your program. If the current link-edit address is not aligned on a page boundary, the linkage editor uses the next higher page boundary address.

Some examples of PHASE statements follow:

```
PHASE PHNAME, **504
```

This causes loading to start 504 bytes past the first doubleword after the end of the partition save area, or past the end of the previous phase.

```
PHASE PHNAME3, PHNAME2
```

This causes loading to start at the same point where the loading of phase PHNAME2 started. When PHNAME2 is also the name of a previous entry point, then loading starts at the address of that entry point, aligned on a doubleword boundary.

```
PHASE PHNAME, CSECT1(PHNAME2)
```

This causes loading to start at the point where CSECT1 was loaded. CSECT1, the named control section, must have appeared in the phase named PHNAME2.

```
PHASE PHNAME1,S+30K  
PHASE PHNAME2,*  
PHASE PHNAME3,PHNAME2
```

The first phase (PHNAME1) of the preceding series is loaded starting at 30K plus the length of the save area. The second phase (PHNAME2) of the series is loaded at the end of PHNAME1. The third phase (PHNAME3) is loaded at the same address as was PHNAME2.

```
PHASE PHNAME,ROOT
```

Loading begins at the first doubleword after the beginning of the partition save area, and the area assigned to the COMMON pool (if any).



---

## Chapter 5. Librarian

This section describes a program, contained in the VSE operating system, which services, copies, and provides access to system and private libraries.

Note that the commands DELETE, MOVE and RENAME cannot be carried out when the specified sublibrary is in use. A sublibrary is "in use" when:

- A LIBDEF job control statement or command specifying that sublibrary is active in another partition, or
- A librarian ACCESS or CONNECT specifying that sublibrary is active in another partition.

---

### Librarian Return Codes

The Librarian program sets a return code each time a librarian function is attempted. If the function completes successfully, a return code of 0 or 2 is set. In the case of non-completion, a return code of 4 to 16 is set, depending on the severity of the error:

#### Return Code

#### Meaning

- |    |  |
|----|--|
| 0  | The command was completed successfully.  |
| 2  | The command was completed successfully and a particular result was reached (for example, for TEST and COMPARE commands). |
| 4  | The command was completed, but an exceptional condition occurred, or the requested result already existed.               |
| 8  | The command was only partly executed, but the librarian program could continue processing.                               |
| 16 | A severe error occurred while processing the command. The librarian program terminates.                                  |

The set of librarian commands includes an ON, a GOTO and a /. label command, which allow conditional execution of following commands dependent on the success of the preceding ones.

The librarian program itself can test the return code after execution of each command. The highest return code set during a librarian call is passed to job control at the end of the librarian job step, and can be tested by conditional job control statements as explained under "The Librarian Program" in the manual *z/VSE Guide to System Functions*.

---

### Library Concept

The librarian program is used to maintain data which must be readily available to the system, such as programs and cataloged procedures. These data are organized in libraries, which are divided into sublibraries, which in turn contain the data, organized in units called members.

Each member is identified by library name, sublibrary name, member name, and member type. Most librarian commands which act on members have an operand in the form "membername.membertype". The "library.sublibrary" component of

## Librarian: Introduction

the name is supplied in a preceding ACCESS or CONNECT command. Library, sublibrary, and member names can be freely chosen, following the rules given in the descriptions of the DEFINE and CATALOG commands.

The member type used depends on the type of data contained in the member, as follows:

**A-Z, 0-9, \$, #, @**

for source programs, which are to serve as input to a language translator;

**OBJ**

for object modules (output from a language translator, input for the linkage editor);

**PHASE**

for executable program phases (output from the linkage editor, ready for loading into storage);

**PROC**

for cataloged procedures;

**DUMP**

for storage dumps.

A "type" other than these five can be used for members containing any user data.

---

## Supported Equipment

The librarian supports libraries on all disk devices which are supported by z/VSE.

---

## Compatibility with Previous Releases

The librarian takes over the functions of the programs MAINT, COPYSERV, CORGZ, CSERV, RSERV, SSERV, DSERV and PSERV, BACKUP and RESTORE, which were supported under earlier releases of VSE.

For compatibility reasons, however, a number of these "old" programs are emulated in the VSE librarian, and can run without change, with the exception of comments coded in the "old" control statements. Any comment in a command submitted to the librarian must be preceded by /\* and followed by \*/. For information on compatibility with and migration from earlier releases, see the chapter "Planning for Migration" in the manual *z/VSE Planning*. Note that the system library is no longer default for the "old" functions, so MAINT, CORGZ, xSERV and LNKEDT jobs must contain a LIBDEF statement for the library to be used. The system logical units SYSxLB are no longer supported by the ASSGN statement or command.

---

## Librarian Command Syntax

The facilities of the librarian are invoked by using the EXEC LIBR statement, followed by one or more librarian commands.

These are in free format, and may be coded anywhere between column 1 and column 72 of the input line. This applies to librarian commands entered from SYSIPT and from SYSLOG.

### Separators

The command name is separated from its operands by one or more blanks.



A comma or blank character is allowed to separate operands or elements of a list. Any of the allowed separators, namely comma, blank, equals sign or colon, two periods, parentheses or comments, may be surrounded by one or more blanks.

### Continuation

Command continuation is indicated by a minus sign (-) as the last non-blank character in a line. This sign is also recognized as a separator, and need not be preceded by a blank, although this is, of course, allowed. The same holds true for comments.

### Comments

Comments are allowed at any place where a blank is allowed. They are identified by a “/” at the beginning and a “\*/” at the end. You may code a comment outside a command, in a line by itself. Do not begin the “/” in column 1, as this would be interpreted as the end-of-file indicator when entered on SYSIPT.

### Abbreviation

Command names may be shortened by leaving off one or more letters from right to left, so long as the name remains unique.

Exceptions to this rule are the commands LISTD, PUNCH and GOTO. LISTD can be shortened to LD. The shortest allowed form of PUNCH is PU. GOTO cannot be shortened.

In the following sections, the part of the command name which must be coded is in capitals, the rest in lower case. For example, you may code the command name given as DEFine in one of the following forms:

DEF, DEFI, DEFIN, or DEFINE.

Operand keywords may be shortened in the same manner, so long as they cannot be confused with other keywords which are valid for the particular command. Here again, you can take the notation of the following sections as a guide.

### Notation

Like all librarian key-words, LIBRARY and SUBLIBRARY can be entered in full or in any shortened form. In the following sections, which describe the librarian commands, this syntax notation is used:

For library specifications: **Lib = l ...**

For sublibrary specifications: **Sublib = l.s ...**

For member specifications: **mn.mt ...**

where

**l** is the library name (1..7 characters),

**s** is the sublibrary name (1..8 characters),

**mn** is the member name (1..8 characters), and

**mt** is the member type (1..8 characters).

Do not use the names IJSYSRS or IJSYSRn for your libraries or the name SYSLIB for a sublibrary. These names are reserved for the SYSRES files and the system sublibrary respectively.

The notation is otherwise the same as that described in “Understanding Syntax Diagrams” on page 2.

## Librarian: Introduction

### Operands

Operands are of two kinds, positional and keyword. Positional operands consist of a value only, for example, **NAMEA.TYPEB**, and must be coded in the position shown in the command description. Keyword operands consist of a keyword and a value separated by an equals sign, for example, **LIST=YES**. These operands may be coded in any order, but they must follow any positional operands in the command.

### Multiple targets

Certain commands can perform the same function on several targets at once. This is indicated in the notation by dots after the operand, for example, **Lib=1** ...

In such a case you can code:

```
Lib=Lib1 Lib2 Lib3
or Lib=Lib1, Lib2, Lib3
or Lib=(Lib1 Lib2 Lib3)
or Lib=(Lib1,Lib2,Lib3).
```

### From-to operands

Commands which require two targets, for example **COMPARE** and **COPY**, where the sequence of the operands determines how the function is to be carried out, can have the operands coded as:

```
Lib=Lib1:Lib2
or Lib=Lib1 Lib2
or Lib=Lib1..Lib2.
```

The same principle holds true when the multiple operands or the two targets are sublibraries or members.

### Generic References

In librarian commands you may make generic reference to member names and types within the sublibrary specified in a preceding **ACCESS** or **CONNECT** command. However, you need **READ** access right to the sublibrary, if it is protected by the Access Control Function.

If you want to specify members of all types with the name **ABC**, code **ABC.\*** for **mn.mt**. Coding **\*.PROC** will cause the command to apply to all members of the type **PROC**, whatever their names may be.

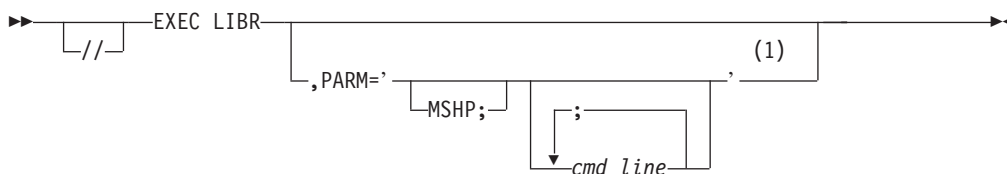
You can specify all members of a given type whose names start with the same combination of characters by coding, for example, **AB\*.OBJ**. This would result in the members **ABC**, **ABEND**, **ABSTAIN**, and so on, being acted upon, provided they have **OBJ** as type.

Coding **\*.\*** makes all members of any type in the specified sublibrary available to the specified command.

---

## Invoking the Librarian Program

Before you issue any librarian commands, you must activate the librarian program using the job control statement or command:



**Notes:**

1 With PARM, you must specify either 'MSHP;' or 'cmd\_line'.

The librarian then accepts commands from SYSLOG, if the EXEC LIBR statement was issued from there, or from SYSIPT. Type in END on SYSLOG, or issue a /\* statement on SYSIPT, after the last librarian command. The librarian then passes a return code to the job control program for use in conditional job control.

In the EXEC command or statement:

**PARM='cmd-line;cmd-line'**

Can be any valid librarian command line. You can code any number of librarian command lines in the PARM operand, enclosed in single quotes and separated by semicolons. However, the number of characters between the quotes must not be greater than 100. This restriction also applies when you enter 'MSHP' followed by librarian commands.

A librarian command of up to 72 characters can be entered in one command line. The normal rules for librarian commands apply (see "Librarian Command Syntax" on page 286).

The command or commands specified in the PARM operand are carried out before any other commands which may follow the EXEC LIBR statement or command.

If the EXEC LIBR statement was issued from SYSIPT and the PARM operand contains CATALOG or UPDATE data, the data must not be a /\* (end-of-file) or /& (end-of-job) string.

Because EXEC LIBR is a job control statement, you can use symbolic parameters in the librarian command lines which you insert in the PARM operand. Be sure, however, that an EXEC LIBR statement with symbolic parameters is called from within a job which starts with a // JOB statement.

**MSHP**

Specifies that members controlled by the Maintain System History Program (MSHP) can be modified by the librarian. Otherwise, these members can be modified only by MSHP.

**Partition Size for the Librarian Program**

The partition in which the Librarian is to run must have at least 256K of virtual storage allocated to it.

For libraries in VSAM managed space, a SIZE specification of at least 256K and sufficient GETVIS space must be given for the partition. This can be given in a SIZE job control statement, or in the SIZE operand of the EXEC LIBR statement.

The specification SIZE=AUTO is not allowed in the EXEC LIBR statement.

---

**Summary of Librarian Commands**

The following is a complete list of the VSE Librarian commands:

## Librarian: Introduction

Command Name	Command Object		
	Library	Sublibrary	Member
ACCESS		X	
BACKUP	X	X	X
CATALOG			X
CHANGE		X	
COMPARE	X	X	X *
CONNECT		X	
COPY	X	X	X *
DEFINE	X	X	
DELETE	X	X	X *
GOTO			
INPUT			
LIST			X *
LISTDIR	X	X	X *
LOCK			X
MOVE	X	X	X *
ON			
PUNCH			X *
RELEASE	X	X	
RENAME		X	X *
RESTORE	X	X	X *
SEARCH	X	X	X *
TEST	X	X	X *
UNLOCK	X	X	X
UPDATE			X
/. label			
/+ End-of Data			
/* or END			

'\*' means: generic specification accepted.

Figure 22. List of Librarian Commands

---

## Merging Sublibraries

There is no MERGE command as such in the Librarian command set. To merge two sublibraries, use a CONNECT command followed by a COPY or MOVE command with a generic member specification.

Using the COPY command leaves the “from” sublibrary intact; using the MOVE command causes the “from” sublibrary members to be deleted.

For examples, see the descriptions of the COPY and MOVE commands.

---

## Librarian Commands

### ACCESS (Specify Target Sublibrary)

►► Access — Sublib=*l.s* —►►  
                   └─?─┘

#### Sublib=*l.s*

Specifies the sublibrary, qualified by the library name, to be used in any

following command which has a member-name.member-type specification as its operand. When given before a RESTORE member command, it specifies the default target sublibrary.

The ACCESS command is valid only if the specified library and sublibrary already exist.

- ? If you code a question mark as operand, the name of the sublibrary currently accessed is displayed on SYSLOG, if the command was entered from there, otherwise on SYSLST.

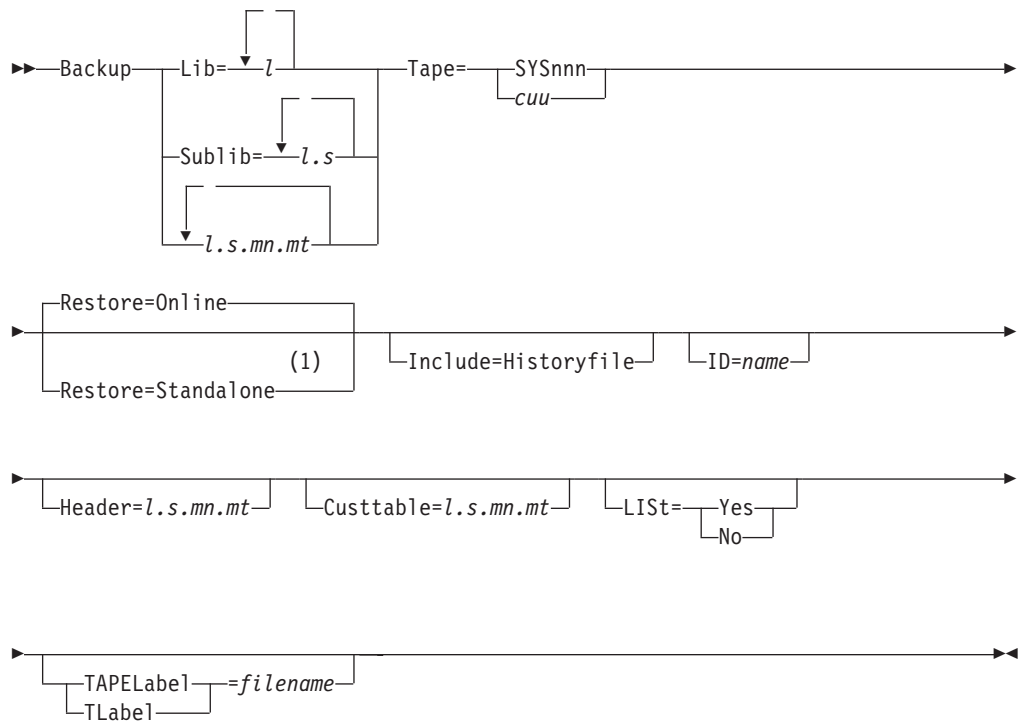
The ACCESS command remains valid until

- Another valid ACCESS command is given, or
- The sublibrary specified in the command is deleted, or
- The sublibrary specified in the command is renamed, or
- The library specified in the command is deleted, either explicitly by a DELETE command, or implicitly by a MOVE, RESTORE or DEFINE command.

The ACCESS command may also become invalid if a DELETE, DEFINE or RENAME for the library/sublibrary specified in the ACCESS command failed.

## BACKUP

### BACKUP (Backup Library or Sublibrary)



#### Notes:

1 Lib=, Sublib=, and l.s.mn.mt are optional if RESTORE is STANDALONE. The BACKUP command causes libraries, sublibraries (including SYSRES files), or members to be copied to tape. If no library, sublibrary, or member is specified, only the stand-alone programs are written onto the output tape and the tape is positioned after the stand-alone utilities file.

The backup function includes a reorganization of the copied library, which usually results in a faster read access after a later restore. The reorganization will be most effective if a complete library is backed up and restored.

#### Lib=l

Specifies the library to be backed up. The library names IJSYSR1 to IJSYSR9 specify SYSRES files, the name IJSYSRS specifies the IPLed system. If you specify IJSYSRS, and no DLBL/EXTENT information is available, the librarian creates a file-id of IJSYSRS.

For this operand you may code a list of library names.

The operand is optional if RESTORE=STANDALONE is specified.

The specification of the DEFINE EXTENTS parameter ( see page 306 ) is written to the backup tape.

#### Sublib=l.s

Specifies the sublibrary to be backed up. For this operand you may code a list of sublibrary names.

The operand is optional if RESTORE=STANDALONE is specified.

**l.s.mn.mt**

Specifies the member to be backed up. For this operand you may code a list of member names. The specification of the member name and the member type may be generic.

The operand is optional if RESTORE=STANDALONE is specified.

**Tape=SYSnnn | cuu**

Specifies the programmer logical unit or the physical unit address of the tape unit to be used for output.

If the amount of data to be backed up requires multiple tapes, you must assign the alternate tape in an ASSGN job control statement. Note, however, that multiple BACKUP commands for unlabeled tapes with alternate assignments may result in the reuse of tapes and the corruption of data in case all assigned units have been exhausted.

If RESTORE=ONLINE is specified, the tape is **not repositioned** before the backup starts.

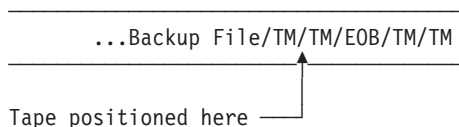
The following files are written to the backup tape:

1. The header (if specified), or empty file.
2. The backup-file-ID record and the system history file (if INCLUDE=HISTORYFILE was specified);
3. The backup file, containing the libraries, sublibraries and members specified in the command.

If RESTORE=STANDALONE is specified, the tape is first positioned at the load point, and then the following files are written to it:

1. The stand-alone IPL file: the header (if specified), the IPL record, and the IPL and supervisor programs.
2. The stand-alone utility file: the stand-alone utility programs and all SVA programs needed in the stand-alone environment.
3. The backup-file-ID record and the system history file (if INCLUDE=HISTORYFILE was specified);
4. The backup file, containing the libraries, sublibraries, and members specified in the command.

At completion of the backup (irrespective of the RESTORE specification), two tape marks, an End-of-Backup (EOB) record, and two more tape marks are written. The tape is left positioned after the tape mark which marks the end of the backup file, thus:



**Restore=Online | Standalone**

Specifies whether the backed up data is to be restored online or stand-alone. The default is ONLINE.

Standalone indicates that SYSRES files (libraries IJSYSR1 to IJSYSR9) can be restored stand-alone; other files (private libraries, sublibraries or members) on the backup tape can be restored Online only.

For a stand-alone restore, the librarian writes an IPL routine onto the backup tape (see above, under **Tape=SYSnnn | cuu**).

For RESTORE=STANDALONE the specification of **lib=**, **sublib=**, or **l.s.mn.mt** is optional. Thus if no library, sublibrary or member is specified on the

## BACKUP

BACKUP command, only the stand-alone programs are written onto the output tape, and the tape is positioned after the stand-alone utilities file.

### **Include=Historyfile**

Specifies that the system history file should also be backed up. It will be copied to tape first, before the specified libraries or sublibraries. The history file must not extend over two tapes.

### **ID=name**

Defines an identification for the backup file to be created by this BACKUP command. This makes it easier to locate a particular version of backed-up data during RESTORE. If this operand is omitted, no identification is recorded.

The **name** must be 1 to 16 characters long, and enclosed in quotes. If only alphanumeric characters are used, the quotes are not needed. **name** itself must not contain quotes.

### **Header=l.s.mn.mt**

Specifies an optional header to be written to the first file on the backup tape. The contents of the header is that of the member specified by l.s.mn.mt, which must have a record length of 80 bytes.

The header may contain, for example, information text, job control statements or copyright statements. It is skipped by the librarian during online restore.

If the backup is to be restored stand-alone, the header on the tape to be IPLed must start with the IPL bootstrap records.

### **Custtable=l.s.mn.mt**

Specifies the phase containing the customization table and the changed message texts for stand-alone Restore and Fastcopy. The contents of the phase is that of the member specified by l.s.mn.mt. This member is written as part of the stand-alone utilities onto the backup tape when RESTORE=STANDALONE is specified. If this operand is omitted, the IBM provided member IJSYSRS.SYSLIB.IJWCUST.PHASE is written onto the backup tape. The operand is ignored if RESTORE=ONLINE is specified.

### **LISt=Yes | No**

If you specify LIST=YES, the names of the restored libraries, sublibraries and (if members were specified in the first operand) members will be printed on SYSLST. This is especially useful if members were specified generically.

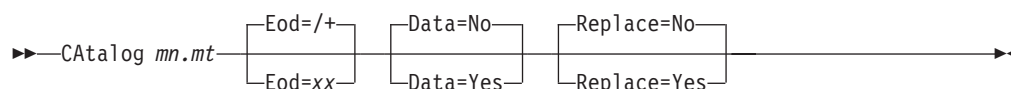
If LIST=NO is specified no listing will be produced. If libraries or sublibraries are specified in the first operand, the default is LIST=YES; if members are specified, the default is LIST=NO.

### **TAPeLabel | TLabel=filename**

Indicates that the tape to be used for output contains standard labels. **filename** is the seven-character file name that you specified in the TLBL statement for your backup file.



## CATALOG (Catalog Member)



The CATALOG command causes the data following it to be cataloged under the name and type specified. The end of the data to be cataloged is usually indicated by a /+ (but see the description of the EOD operand).

An empty data set will not be cataloged.

Any member types (except PHASE and DUMP) can be cataloged into any sublibrary using this command.

If you wish to add, delete or replace lines in the member at a later date, using the librarian UPDATE command, you should provide sequence numbering in your input lines. The sequence number may be located anywhere in the line, and may be 1 to 8 characters long. The format must be:

- Numeric, with leading zeros, if you want to update the member using SEQUENCE=*n* or SEQUENCE=FS on the UPDATE command;
- Alphanumeric, without blank characters, if you want to update the member using SEQUENCE=NO on the UPDATE command.

The librarian provides the necessary sequence numbering if you use the following sequence of commands:

EXEC LIBR	call the librarian program
ACCESS S=lib.sublib	access appropriate sublibrary
UPDATE membername.membertype	"update" the member
)END	end update, causing renumbering

These commands cause the lines of the specified member to be sequence numbered in columns 77 to 80 using an increment of 10, starting at 10. These are the default values used by the librarian. Should you wish to use a different increment or a different length or position for your sequence numbering, specify the appropriate values in the SEQUENCE and COLUMNS operands of the UPDATE command.

### **mn.mt**

Specifies the name and type under which the following data are to be cataloged. *mn* and *mt* may each be 1 to 8 characters long, and must be alphanumeric. The member type must not be PHASE or DUMP, because members of these types can be cataloged only by the Linkage Editor and Dump program respectively.

The sublibrary in which the member is to be cataloged must have been created, and must have been specified in a preceding ACCESS command, before you attempt to catalog a member into it.

If a member of the specified name and type existed in the target sublibrary before the command was carried out, its "put-in-sublibrary" time stamp is left unchanged, and the "last-replaced" time stamp is updated.

If such a member **did not** exist in the target sublibrary, the "put-in-sublibrary" time stamp is set, and the "last-replaced" time stamp is empty.

## Eod=xx

Specifies what combination of two characters is to be used to indicate end-of-data in the following input. The default is /+. You must use an alternate end-of-data delimiter

- When the data to be cataloged contains a /+ statement, and
- When the CATALOG command and its input data are part of a job control procedure. (If a /+ were included in the SYSIPT data of a procedure, the job control program would take it as the End-of-Procedure delimiter, and not process the remainder of the procedure.)

For xx you may code any two characters except /\*, comma, blank or minus sign. For further information, see the /+ command on page /+ (End-of-Procedure) on page 253.

Except when cataloging members of the type OBJ or assembler macros with a MEND statement, the end-of-data record must follow the last record of the input data. The end-of-data record itself is not cataloged.

## Data=No | Yes

Is applicable only for procedures. The member type must be PROC or a user type. You must code DATA=YES if the procedure contains SYSIPT data; in that case, **all** the data must be included within the procedure. (In a cataloged procedure, to read data on SYSIPT, you must use a DTFDI instead of a DTFCF.) The system default is DATA=NO. Procedures to be used in a set of nested procedures must be cataloged either all with DATA=YES or all with DATA=NO.

## Replace=No | Yes

Allows conditional cataloging. If you specify REPLACE=YES, an already existing member with the same name and type as that specified will be deleted and overwritten with the new data, that is, it will be replaced by the new member. However, only members which are **not locked** will be replaced by new members. (To unlock a member, use the UNLOCK command.)

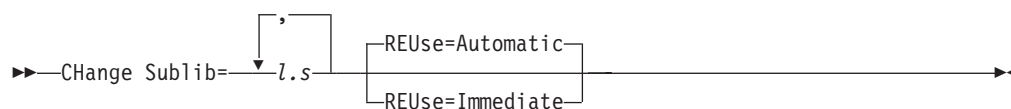
REPLACE=NO, the default, means that the input data will not be cataloged if a member with the same name and type already exists in the sublibrary.

For compatibility with the old CATALOG function, BKEND, MACRO and MEND statements are allowed instead of the EOD statement. They can start in column 1. The Librarian EOD indication may be specified in addition. It must follow the BKEND or MEND statement immediately. If a member starts with BKEND or MACRO, its end must be indicated by the corresponding BKEND or MEND statement.

A macro with input data following it must be cataloged with the following delimiting statements:

```
BKEND / MACRO / macro source statements / MEND / input data / BKEND
```

## CHANGE (Change REUSE Attribute)



The CHANGE command can be used to change the REUSE attribute of a sublibrary. For details on the REUSE attribute, see the DEFINE command.

### **Sublib=l.s**

Specifies the qualified name of the sublibrary whose REUSE attribute is to be changed. You may specify several sublibraries which are to have the same attribute.

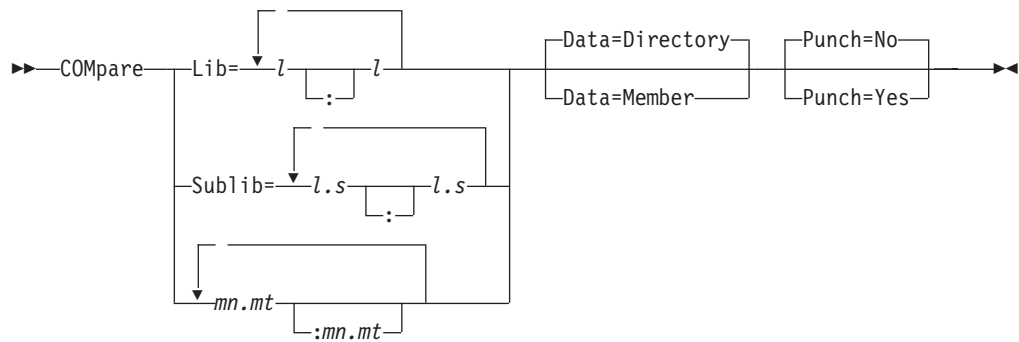
### **REUse=Automatic | Immediate**

Specifies which REUSE attribute the specified sublibrary (or sublibraries) should have from now on. If the sublibrary already has the specified attribute, the librarian program sets a return code of 4.

When IMMEDIATE is specified, any space which is no longer in use in the sublibrary, but has not yet been freed, is freed at once.

## COMPARE

### COMPARE (Compare Libraries, Sublibraries or Members)



The COMPARE command is used to compare libraries, sublibraries or members, and provide a listing of the differences, as explained under the DATA operand below. Comparison can be made between the member names or the member contents. Any locking information will be ignored. Only elements which are present in the first library entity are compared.

If the entities being compared are not equal, the librarian issues message L008I and sets a return code of 2.

#### Lib=l

Specifies the names of the libraries to be compared. Comparing is done for all sublibraries in each of the specified libraries. If a sublibrary corresponding to the current sublibrary in the first library is not found in the second library, a message is issued, and comparing continues with the next sublibrary.

#### Sublib=l.s

Specifies the sublibraries to be compared. An equals sign can be coded in the second operand in place of a name which is identical to one in the first operand, for example, instead of:

```
LIB1.SUBLIBA : LIB2.SUBLIBA
```

you can code:

```
LIB1.SUBLIBA : LIB2.=
```

#### mn.mt [ : mn.mt ...]

Specifies the member(s) to be compared. The member specified first (source member) is compared with the member specified second (target member). The sublibraries in which the source and the target members reside must have been specified in a preceding CONNECT command. If only the source member is specified, it is assumed that source and target member name and type are identical. If a target specification is to be identical with the source specification, this can be indicated with an equal (=) sign:

```
A.B : C.=  
gives A.B : C.B
```

If the member specifications are generic, the *target* member specification is built in the following manner: Every non-generic string that fits into a generic one is divided into two parts: the known part and the rest string. For example: ANTON fits into A\*. 'A' is the known part and 'NTON' the rest part. The target string is built by concatenating the known part of the target string with the rest part of the source string. For example:

Generic specification: A\*.A : B\*.A  
 Results in: ANTON.A : BNTON.A

Generic specification: AN\*.A : B\*.A  
 Results in: ANTON.A : BTON.A

The known part of the target specification must be smaller or equal in length as compared to the known part of the source specification. This ensures that the target member names and types do not exceed eight characters in length.

#### **Data=Directory | Member**

Specifies whether the directories or the member contents are to be compared. The default is DATA=DIRECTORY.

If Directory is specified or the operand is omitted, the names and types of the members which reside in the first library or sublibrary but not in the second are printed on SYSLST.

If Member is specified, the contents of the members of the specified name and type are compared record by record. Comparing stops for a member when the first mismatch is found, and the mismatching records are printed on SYSLST.

If a member is not found in the second sublibrary, a message is issued, and comparing continues with the next member, if any.

#### **Notes:**

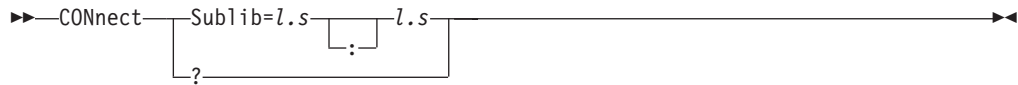
1. Only members with the same internal representation can be compared. Phases have a different internal representation from other member types.
2. Phases linked in different partitions have different starting addresses, and give a mismatch.

#### **Punch=Yes | No**

This operand is valid only if you specify DATA=DIRECTORY. If you specify PUNCH=YES, the system generates COPY statements on SYSPCH. The operands of these COPY statements are the names of the members found in the first sublibrary but not in the second sublibrary.

The Librarian program writes an EOF on SYSPCH at completion of the COMPARE function.

## CONNECT (Specify 'From' and 'To' Sublibraries)



The CONNECT command must be used before COPY, MOVE, or COMPARE commands which have a member specification as operand. It provides the names of the sublibraries in which the members are to be found or placed.

The function is similar to that of the ACCESS command, except that CONNECT must be used before commands which require two sublibraries to be specified.

### Sublib=l.s:l.s

The first operand of the CONNECT command specifies the “from” or first sublibrary required for the following commands, and the second operand the “to” or second sublibrary. Both sublibraries must exist before execution of the command.

If a name in the second operand is the same as a name in the first operand, an equals sign can be coded in place of it in the second operand. For example, instead of:

```
CON S=LIB1.SUBLIBA : LIB1.SUBLIBB
```

you can code;

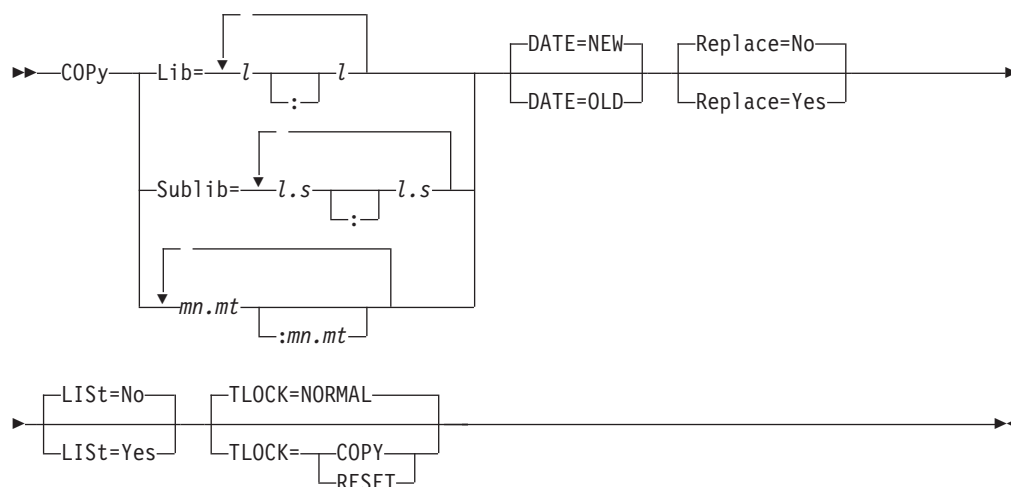
```
CON S=LIB1.SUBLIBA : =.SUBLIBB
```

- ? If a question mark is specified as operand, the current CONNECT information is displayed on SYSLOG, if the command was entered from there, otherwise on SYSLST.

The CONNECT command remains valid until

- Another valid CONNECT command is given
- A sublibrary specified in the command is deleted
- A sublibrary specified in the command is renamed
- A library specified in the command is deleted, either explicitly by a DELETE command, or implicitly by a DEFINE, MOVE or RESTORE command.

## COPY (Copy Library, Sublibrary or Member)



The COPY command is used to copy libraries, sublibraries or members. Copying is allowed between any supported disk devices. COPY will usually not delete or replace locked members in the target library, unless otherwise specified in the TLOCK operand.

This command can also be used to merge sublibraries. To do this, use a command sequence like the following example:

```
CONNECT SUBLIB = LIBA.SUB1 : LIBB.SUB2
COPY    *.*
```

LIBB.SUB2 will now contain all the members previously present in either of the two specified sublibraries. Members of the same name and type which were present in both sublibraries before the merge, will remain as they were in the “to” sublibrary.

If the common members are to have the same content in the merged sublibrary as they had in the “from” sublibrary, use a command sequence like the following example:

```
CONNECT SUBLIB = LIBA.SUB1 : LIBB.SUB2
COPY    *.* REPLACE=YES
```

### Lib=l:l

Specifies the libraries to be used in the copy operation. All sublibraries of the library specified first are copied to the library specified second. The to-library must exist before a copy operation can be started.

### Sublib=l.s:l.s

Specifies the sublibraries to be used in the copy operation. The sublibrary specified first is copied into the library specified second, where it is stored under the sublibrary name specified with the second library. It is therefore possible to change the name of the sublibrary while copying it.

The REUSE attribute of the target sublibrary is retained, if this sublibrary existed. Otherwise, the attribute of the “from” sublibrary is used.

If a name in the second operand is the same as one in the first operand, it can be replaced by an equals sign. For example, instead of

```
LIB1.SUBLIBA : LIB2.SUBLIBA
```

you can code

```
LIB1.SUBLIBA : LIB2.=
```

#### **mn.mt [ : mn.mt ...]**

Specifies the member(s) to be copied. The member specified first (source member) is copied into the member specified second (target member). The sublibraries in which the source and the target members reside must have been specified in a preceding CONNECT command. If only the source member is specified, it is assumed that source and target member name and type are identical. If a target specification is to be identical with the source specification, this can be indicated with an equal (=) sign:

```
      A.B : C.=
gives A.B : C.B
```

If the member specifications are generic, the *target* member specification is built in the following manner: Every non-generic string that fits into a generic one is divided into two parts: the known part and the rest string. For example: ANTON fits into A\*. 'A' is the known part and 'NTON' the rest part. The target string is built by concatenating the known part of the target string with the rest part of the source string. For example:

```
Generic specification: A*.A      : B*.A
Results in:          ANTON.A    : BNTON.A
```

```
Generic specification: AN*.A     : B*.A
Results in:          ANTON.A    : BTON.A
```

The known part of the target specification must be smaller or equal in length as compared to the known part of the source specification. This ensures that the target member names and types do not exceed eight characters in length.

#### **DATE=NEW | OLD**

DATE=NEW indicates that when you copy a library, sublibrary, or member, the creation dates in their time stamps are set to the actual date and time of the copy operation. Likewise, all members in the library or sublibrary receive time stamps of the actual date and time: the CREATION DATE field shows the date and time of the copy operation.

DATE=OLD indicates that the date and time of the source item is to be used. If you specify DATE=OLD for a member you want to copy, the CREATION DATE or LAST UPDATE time stamp might be older than the creation date of the sublibrary or library. If you specify DATE=OLD for a sublibrary you want to copy, the copied sublibrary may have an earlier creation date than the containing library.

#### **Replace=Yes | No**

Specifies whether copying should be conditional or unconditional.

For **unconditional** copying, specify REPLACE=YES. This means that:

- With COPY LIB=..., all sublibraries of the from-library are copied into the to-library. In the to-library, any existing sublibraries which have the same names as copied sublibraries are overwritten. Any existing sublibraries of the to-library which are not overwritten by copied sublibraries remain in the to-library;
- With COPY SUBLIB=..., the to-sublibrary is first emptied, if it already exists, or created, if it does not yet exist. Then all members of the from-sublibrary are copied into the to-sublibrary;



- With COPY mn.mt..., the specified member of the from-sublibrary is copied into the to-sublibrary. If a member of the specified name and type exists in the to-sublibrary, it is overwritten.

For **conditional** copying, specify REPLACE=NO or omit the REPLACE operand. This means that:

- With COPY LIB=..., a sublibrary of the from-library is not copied if a sublibrary of the same name exists in the to-library. All other sublibraries of the from-library are copied, and all existing sublibraries of the to-library remain in that library;
- With COPY SUBLIB=..., the from-sublibrary is not copied if the to-sublibrary already exists;
- With COPY mn.mt..., the specified member is not copied if a member of the same name and type exists in the to-sublibrary.

When you specify members generically, each from-sublibrary member which matches the specification is compared singly with existing members in the to-sublibrary. Therefore, some of the specified members may be copied, while others are not.

#### **LISt=Yes | No**

If YES is specified, the names and types of the members copied and those of the corresponding “to” and “from” libraries and sublibraries will be printed on SYSLST.

This is especially useful if the members were specified generically or in the case of conditional copying, with REPLACE=NO.

#### **TLOCK=COPY | RESET | NORMAL**

Controls the locking status of the copied member in the target sublibrary.

##### **Target member does not exist:**

- TLOCK=COPY indicates that the target member gets the locking status of the source member.
- TLOCK=RESET or NORMAL indicates that the target member will be unlocked.

##### **Target member exists:**

If REPLACE=NO was specified, any library member or sublibrary that already exists in the to-library is not copied and therefore keeps its locking status, irrespective of the specified TLOCK operand.

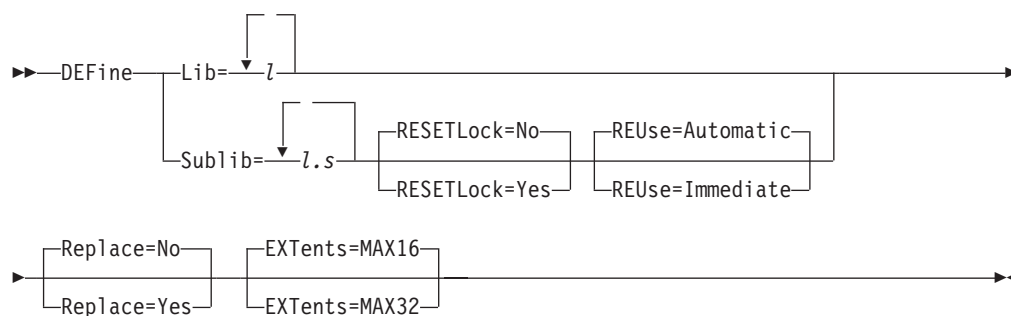
If REPLACE=YES was specified,

- TLOCK=COPY indicates that the locking status of the source member is to be copied to the target member, that is, the processed target member will be (un)locked if the source member was (un)locked. A locked member will be replaced.
- TLOCK=RESET indicates that the specified target member(s) will receive a lock status of ‘unlocked’, no matter whether the source member was locked or unlocked. The member will be copied in any case and the target will be replaced.
- TLOCK=NORMAL indicates the following:
  - If a whole library is to be copied, the command will bypass those target sublibraries that contain locked members. All copied members will be unlocked in their target.

## **COPY**

- If a sublibrary is to be copied and the target sublibrary contains locked members, this sublibrary will not be copied.
- If one or more members are to be copied and a target member already exists which is locked, that member is not copied (and a message is issued).

## DEFINE (Define Library or Sublibrary)



The DEFINE command is used to create system libraries (SYSRES files), private libraries and sublibraries.

### Lib=l

Specifies the name(s) of the library or libraries to be created. Library names may be 1 to 7 characters long, and must be alphanumeric. The first character must be alphabetic. Use the names IJSYSR1 to IJSYSR9 to define SYSRES files to be created. The name IJSYSRS must not be used, as it defines the IPLed system.

The device type for SYSRES files is independent of the device type of the IPLed system. The physical location of the library must be specified in DLBL and EXTENT statements. (For a library in VSAM managed space, only the DLBL statement is required.)

If a library is to be created in BAM managed space, the EXTENT statement must have a VOLID and/or logical unit specification.

If a private library is in VSAM managed space, the library space must have been defined previously as (VSAM-) non-reusable with an Access Method Services DEFINE CLUSTER command.

The library name in the DEFINE command must be the same as the filename in the DLBL statement. The type code on the DLBL statement must be SD (default) or VSAM.

A library may consist of more than one extent on different volumes, in which case the disk types must be the same. A library cannot, however, be defined on a split-cylinder extent. The minimum size for a private library extent is 1 track on CKD devices or 10 blocks on FBA devices. The maximum number of extents is 16.

For a SYSRES file, only one extent is allowed, and it may not be in VSAM managed space. On the EXTENT statement, specify track 1 as start address and 2 as the minimum number of tracks on CKD devices. On FBA devices, you must specify block 2 as start address and 28 as the minimum number of blocks in the EXTENT statement.

**Note:** SDL must not be used as a library name, as it is a reserved keyword for the System Directory List.

### Sublib=l.s

Specifies the qualified name(s) of the sublibrary or sublibraries to be created. Sublibrary names may be 1 to 8 characters long, and must be alphanumeric.

## DEFINE

Sublibraries are not allocated a fixed amount of space; their size at a given time is the sum of the sizes of their members.

A library can have any number of sublibraries.

### **RESETLock=No | Yes**

For `REPLACE=YES`, `RESETLOCK=NO` causes the existing sublibrary to be deleted only if it contains no locked members. `RESETLOCK=YES` causes the specified sublibrary to be deleted in any case.

`REPLACE=NO` has no effect on the `RESETLOCK` operand.

`RESETLOCK` is ignored for `DEFINE LIB`. Therefore, `REPLACE=YES` causes an existing library to be deleted even if it contains locked members.

### **REUse=Automatic | Immediate**

This operand is needed in a disk sharing environment, or when several tasks share the same sublibrary. It specifies how the space occupied by deleted members is to be freed. `REUSE=IMMEDIATE` causes the space to be freed as soon as the members are deleted. `REUSE=AUTOMATIC` causes the space to be freed only when the sublibrary is in use by only one task in a non-shared environment; in a shared environment the space can only be freed by issuing the `RELEASE` command.

### **Replace=No | Yes**

This operand controls conditional creation of libraries and sublibraries. If you specify `NO`, or omit the operand, no new library or sublibrary will be defined if one with the specified name already exists.

If you specify `YES`, any previously defined library or sublibrary is overlaid by the new one. That is, the “to” library or sublibrary is deleted implicitly. If the “to” sublibrary is on a disk volume shared by two or more CPUs, the system issues a message prompting the operator to verify this deletion.

Note that, even if `REPLACE=YES` is specified, the sublibrary will **not** be re-defined if one with the specified name exists and is **in use**.

### **EXTents=MAX16 | MAX32**

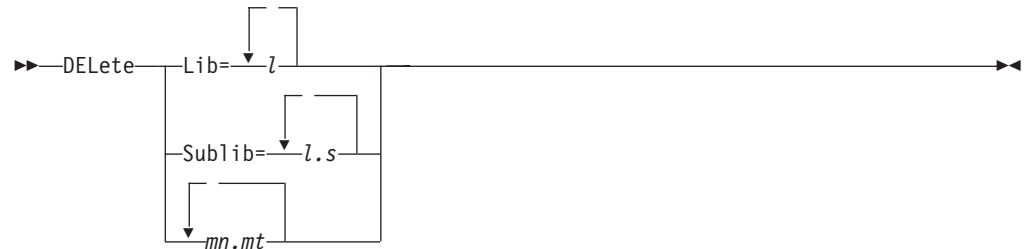
Specifies the extent limit for each library. Note that for BAM libraries a maximum of 16 extents is used even if 32 extents had been specified. The maximum of 32 extents leads to a small additional amount of 24 bit system GETVIS storage. Note that if the `EXTents` parameter is changed for a library, a `VSAM DEFINE/DELETE CLUSTER` command is needed to prevent inconsistencies between the number of VSAM extents and used librarian extents.

Perform the following steps to change the maximum extents of a library:

1. `DEL L=lib`
2. `VSAM DELETE CLUSTER`
3. `VSAM DEFINE CLUSTER`
4. `DEF L=lib EXT=MAXxx`

You can only use a shared library with more than 16 extents if all systems accessing this library can handle more than 16 extents. The `EXTents` parameter must not be used for a library which is shared with systems prior to VSE/ESA 2.5.

## DELETE (Delete Library, Sublibrary or Member)



The DELETE command is used to delete members, sublibraries or libraries.

If a library or sublibrary contains **locked** members, the delete request for this library/sublibrary will be ignored. If a list of members is specified, the delete request will be bypassed for the locked members. (Members can be deleted after they have been unlocked.)

A sublibrary can not be deleted if it is in use in another VSE partition at the time the DELETE command is entered. If a library or sublibrary has been specified in a LIBDEF statement, the LIBDEF definition has to be dropped or changed accordingly before the DELETE command is entered. Any ACCESS or CONNECT commands which were issued for the deleted sublibraries are dropped.

If the library or sublibrary specified in a DELETE command is on a disk volume shared by two or more CPUs, the system issues a message prompting the operator to verify the deletion.

### **Lib=l**

Specifies the library or libraries to be deleted. The system library IJSYSRS may not be deleted, as it contains the IPLed system. The DELETE command causes only the corresponding VTOC entries to be removed. A library in VSAM managed space can be deleted physically only by using the VSAM Access Method Services DELETE command.

### **Sublib=l.s**

Specifies the sublibrary or sublibraries to be deleted. The system sublibrary IJSYSRS.SYSLIB cannot be deleted.

### **mn.mt**

Specifies the member(s) to be deleted. The sublibrary has to be specified in a preceding ACCESS command.

## GOTO

### GOTO (Skip to Label)

▶▶—GOTO *label*—————▶▶

The Librarian GOTO command has the same function as the job control GOTO command. It causes the Librarian to skip all librarian commands up to the **librarian** /. label command specified. You must not specify a **job control** /. label statement.

#### **label**

Specifies the operand of the /. label command after which processing is to continue. The first /. label command with the specified operand is taken, even if there are several with the same operand. The specified /. label command must come after the GOTO command in the command stream.

#### **Notes:**

1. The command name GOTO cannot be shortened.
2. The GOTO command is ignored if entered from SYSLOG.

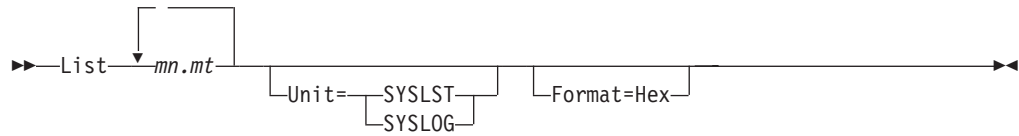
## INPUT (Read from SYSIPT)

▶▶—Input SYSIPT—————▶▶

The INPUT command causes the librarian to read any following commands from SYSIPT instead of SYSLOG until the end of the current job step. This enables you, for example, to enter librarian commands at SYSLOG and data to be cataloged on SYSIPT.

The command has no effect if SYSIPT is already defined as the input device.

## LIST (List Member Contents)



The LIST command causes the contents of one or more members to be displayed on SYSLST or SYSLOG. The LIST command is not intended for printing dumps. See “Dump Analysis Examples” in the manual *z/VSE Guide for Solving Problems*, to perform this.

Phases and dumps are listed in a combined hexadecimal and character string format.

### **mn.mt**

Specifies the member(s) to be displayed. Specify the sublibrary in a preceding ACCESS command.

### **Unit=SYSLST | SYSLOG**

Specifies the output device to be used. If the LIST command is issued from SYSLOG, the default output device is also SYSLOG. If the command is issued from SYSIPT, the default output device is SYSLST.

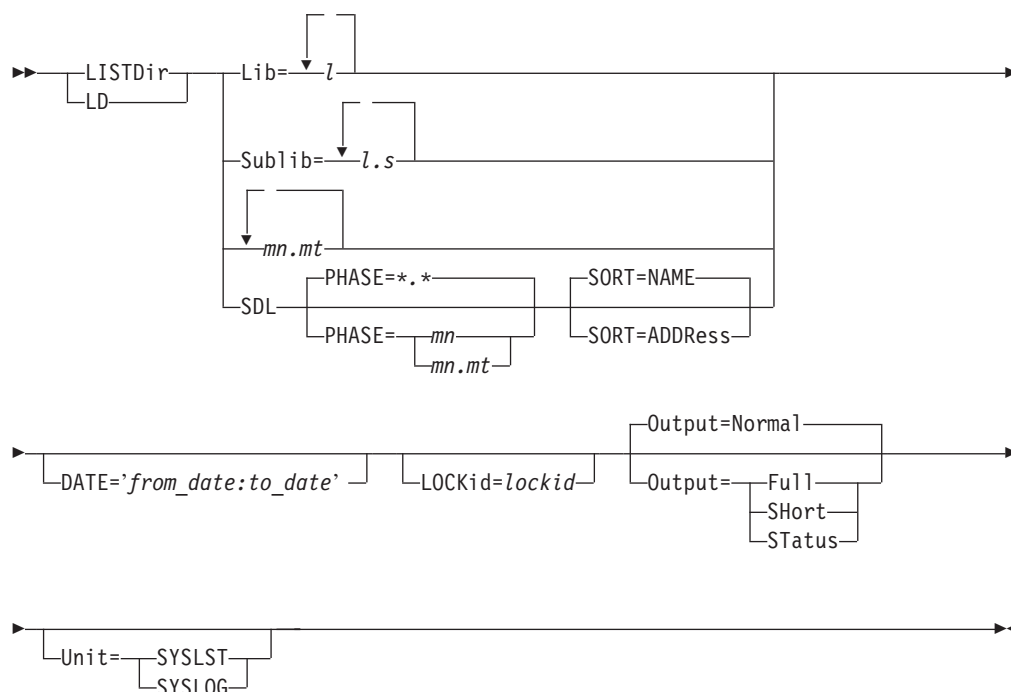
**Note:** If the output is on SYSLOG, only the first 68 positions of each line are displayed.

### **Format=Hex**

Has no effect on the output of phases and dumps. For all other members, specifying FORMAT=HEX results in the character string representation of each record of a member being followed by a two-line hexadecimal translation.



## LISTDIR (List Directory Information)



The LISTDIR (list directory) command is used to display the contents of a directory. The output is a list sorted in alphanumeric collating sequence.

This command can be used to check whether an entity (library, sublibrary or member) exists. If the specified entity does not exist, the librarian sets a return code of 4. This can mean nothing else when set for LISTDIR, so the non-existence of the entity can be tested by using the ON command.

**Note:** If the LISTDIR command is processed, while a sublibrary is being updated from another partition, the resulting output may be incorrect. There may be discrepancies between the library directories and the sublibrary information, or between the sublibrary directories and the member information.

### Lib=l

Specifies that the directory information of a library or libraries is to be displayed. The information provided includes the directory contents of all sublibraries in the specified libraries.

### Sublib=l.s

Specifies that the directory contents of a sublibrary or sublibraries is to be displayed. For sublibrary directory listings, the primary sort field is the member type, so that the members are grouped by type.

### mn.mt

Specifies one or more members. This causes the librarian to display only those parts of the sublibrary directory which are relevant to the named member(s). The sublibrary to be used must be specified in a preceding ACCESS command.

## LISTDIR

### SDL

Specifies that the system directory list is to be displayed. The DATE, LOCKID, and OUTPUT=Full operands are not applicable when SDL is specified.

### PHASE=\*,\* | mn | mn.mt

Specifies that SDL output is to be displayed selectively for one or more phases.

### SORT=NAME | ADDRESS

Specifies that SDL output is sorted by the phase name (default) or by the address in the SVA. If SORT=ADDRESS is specified, only the phases that are loaded into SVA are displayed.

### DATE='yyyy/mo/dd-hh.mi:yyyy/mo/dd-hh.mi'

Indicates that member directory information is to be displayed only for those members that were created or updated during the specified time interval (from:to). There must be no blank between the colon and the specified time values.

#### yyyy

year

**Note:** Due to the Year 2000 changes, the parameter DATE (LISTDIR) accepts 2-digit-, 3-digit-, and 4-digit-year specifications. No leading blanks or zeros must be specified.

#### mo

month (01 - 12)

#### dd

day (01 - 31)

#### hh

hour (00 - 24)

#### mi

minute (00 - 59 if hh<24, 00 if hh=24)

For members, the time stamp that is being checked is the date of the last update. If a member has never been updated, the original time stamp will be checked. For libraries and sublibraries the operand does not apply. Also, the operand may not be specified together with SDL.

It is possible to specify *one* time value only (either the 'from-value' or the 'to-value'). For every date/time value, at least the year must be specified. Omitted values are assumed to last either from the beginning or to the end of the specified year, month, day or hour.

To specify the 'to-value' a semicolon (;) must be placed in front of the time value.

To determine a time value, the Librarian uses the sliding window mechanism. This means, that 79 has to be deducted from the current year to determine the start ('from-value') of the window, whereas 20 has to be added to determine the end value ('to-value') of the window. Every specified year must lie within the window's time span.

If a 2-digit-year is specified, the default sliding window will calculate internally a 4-digit-year, thus indicating the century, too (see example below). If a 3-digit- or 4-digit-year lies too far in the past or the 'from-value' is not specified at all, the lower limit value of the window will be used (January 1, 00:00 for a missing specification). If the chosen year lies too far in the future or the 'to-value' is not specified, the upper limit value of the sliding window will be used (December 31, 24:00 for a missing specification).

## Examples:

DATE='02' (Assuming the current year is 1997, 02 results in 2002, as it lies between the two limit values of the sliding window: 1997-79=1918 and 1997+20=2017. The date means from 2002/01/01 on.)

DATE=':3333' (Assuming the current year is 1997, the upper limit value of the sliding window will be used, because 3333 exceeds the window's time span. Consequently, this date means from 1918/01/01 until 2017/12/31.)

DATE='1991/06/09:1992' (meaning from ... to 1992/12/31)

DATE='1991:1992' (meaning from 1991/01/01 to 1992/12/31)

DATE='1990/12/05-23.30' (meaning from ... until 2017/12/31, assuming 1997 is the current year)

DATE=':1992/04/02-12' (meaning from 1918/01/01 until ..., assuming 1997 is the current year)

**LOCKID=lockid**

Specifies that information is to be displayed only for library members locked with the specified lock identifier. **lockid** is a string of up to eight alphanumeric characters; it can also be generic.

For example, the output of the LISTDIR...LOCKID=BILL,OUTPUT=FULL command will then show, under 'Member Information':

```
LOCKED      : YES
LOCKID      : BILL
```

If OUTPUT=SHORT has been specified, the display contains no locking information.

For OUTPUT=NORMAL, the lockid is displayed instead of AMODE and RMODE information.

The operand may not be specified together with SDL.

**Output=Full | Normal | Short | Status**

Controls the kind and amount of information provided. The specifications FULL, NORMAL and SHORT are applicable for libraries, sublibraries and members. STATUS is applicable only for libraries, sublibraries, and SDL. For SDL, the following OUTPUT parameters are allowed: NORMAL, SHORT, and STATUS. OUTPUT=NORMAL is the system default.

For OUTPUT=FULL, the output contains locking information of the members, for example:

```
LOCKED      : YES      or      LOCKED      : NO
LOCKID      : SAFE     LOCKID      : -
```

**Unit=SYSLST | SYSLOG**

Specifies the output device to be used. If the LISTDIR command is issued from SYSLOG, the default output device is also SYSLOG. If the command is issued from SYSIPT, the default output device is SYSLST.

## LOCK (Lock Member)

▶▶—LOCK *mn.mt* LOCKid=*lockid*—▶▶

The LOCK command causes a library member to be locked for any write or update access (if, for example, a user at a workstation wants to edit a member). The member can be unlocked with the UNLOCK command.

The LOCK command will lock the specified member with the specified **lockid** unless

- The member is already locked,
- You have no UPDATE access right for the member.

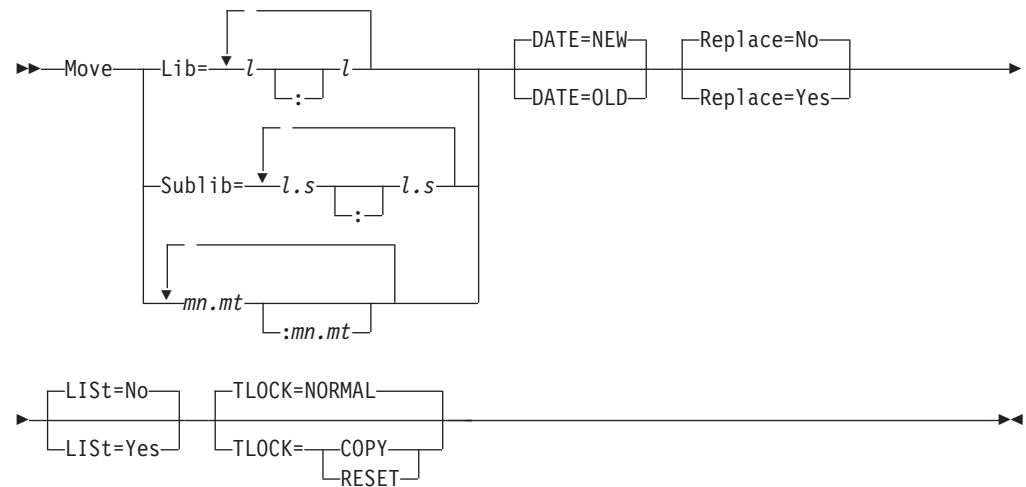
### **mn.mt**

Specifies the member name and member type of the member to be locked. No generic specification is allowed. The sublibrary of the member must be specified in a previous ACCESS command.

### **LOCKid=lockid**

Specifies a lock identifier which should be chosen uniquely for every user. The **lockid** is a string of up to eight alphanumeric characters; no generic specification is allowed. The specified lockid must be used if the library member is to be unlocked again (with a corresponding UNLOCK command).

## MOVE (Move Library, Sublibrary or Member)



The MOVE command works in a similar way to the COPY command, except that the data which have been moved to a target library or sublibrary are deleted from the from-location after they have been copied. When a sublibrary is moved, either explicitly or as part of a moved library, any ACCESS or CONNECT commands for the “from” sublibrary are dropped.

MOVE will never delete locked members in the source library. It will usually not replace locked members in the target library, unless otherwise specified in the TLOCK operand.

Libraries, sublibraries, or members can be moved.

As the moving of a sublibrary causes an implicit deletion of the “from” sublibrary, the MOVE function will not be executed if the “from” library:

- has been specified in a previous LIBDEF statement, or
- is in use by another VSE partition or task at the time the MOVE command is entered.

If the sublibrary specified in a MOVE command is on a disk volume shared by two or more CPUs, the system issues a message prompting the operator to verify the implicit deletion.

The system sublibrary IJSYSRS.SYSLIB cannot be moved.

This command can also be used to merge sublibraries. To do this, use a command sequence like the following example:

```
CONNECT SUBLIB = LIBA.SUB1 : LIBB.SUB2
MOVE      *.*
```

LIBB.SUB2 will now contain all the members previously present in either of the two specified sublibraries. Members of the same name and type which were present in both sublibraries before the merge, will remain as they were in the “to” sublibrary.

## MOVE

If the common members are to have the same content in the merged sublibrary as they had in the “**from**” sublibrary, use a command sequence like the following example:

```
CONNECT SUBLIB = LIBA.SUB1 : LIBB.SUB2
MOVE    *.*    REPLACE=YES
```

This will leave the “from” sublibrary empty.

### **Lib=l:l**

Specifies the libraries to be used in the MOVE function. All sublibraries of the library specified in the first operand will be moved to the library specified in the second operand. The to-library must exist before execution of the MOVE function.

The “from” library still exists after the MOVE, but it is empty if all its sublibraries are copied. The latter is always the case when REPLACE=YES is specified.

### **Sublib=l.s:l.s**

Specifies the sublibraries to be used in the MOVE function. The sublibrary specified in the first operand will be moved to the library specified in the second operand, where it will reside under the sublibrary name specified in the second operand. The REPLACE operand specifies whether an already existing sublibrary of the same name in the to-library is to be overwritten or not. Any ACCESS or CONNECT commands for the “from” sublibrary are dropped.

The REUSE attribute of the target sublibrary is retained, if this sublibrary existed. Otherwise, the attribute of the “from” sublibrary is used.

### **mn.mt [ : mn.mt ...]**

Specifies the member(s) to be moved. The member specified first (source member) is moved into the member specified second (target member). (Use the REPLACE operand to control the deletion of members with duplicate names in the target sublibrary.) A MOVE into the same sublibrary will be rejected, since this can be done more effectively with the RENAME command.

The sublibraries in which the source and the target members reside must have been specified in a preceding CONNECT command. If only the source member is specified, it is assumed that source and target member name and type are identical. If a target specification is to be identical with the source specification, this can be indicated with an equal (=) sign:

```
      A.B : C.=
gives A.B : C.B
```

If the member specifications are generic, the *target* member specification is built in the following manner: Every non-generic string that fits into a generic one is divided into two parts: the known part and the rest string. For example: ANTON fits into A\*. ‘A’ is the known part and ‘NTON’ the rest part. The target string is built by concatenating the known part of the target string with the rest part of the source string. For example:

```
Generic specification: A*.A      : B*.A
Results in:          ANTON.A    : BNTON.A
```

```
Generic specification: AN*.A     : B*.A
Results in:          ANTON.A    : BTON.A
```

The known part of the target specification must be smaller or equal in length as compared to the known part of the source specification. This ensures that the target member names and types do not exceed eight characters in length.

If a member of the specified name and type existed in the target sublibrary before the command was carried out, its “put-in-sublibrary” time stamp is left unchanged, and the “last-replaced” time stamp is updated.

If such a member **did not** exist in the target sublibrary, the “put-in-sublibrary” time stamp is set, and the “last-replaced” time stamp is empty.

#### DATE=NEW | OLD

DATE=NEW indicates that when you move a library, sublibrary, or member, the creation dates in their time stamps are set to the actual date and time of the move. Likewise, all members in the library or sublibrary receive time stamps of the actual date and time: the CREATION DATE field shows the date and time of the move operation.

DATE=OLD indicates that the date and time of the source item is to be used. If you specify DATE=OLD for a member you want to move, the CREATION DATE or LAST UPDATE time stamp might be older than the creation date of the sublibrary or library. If you specify DATE=OLD for a sublibrary you want to move, the moved sublibrary may have an earlier creation date than the containing library.

#### Replace=Yes | No

Specifies whether moving should be conditional or unconditional.

For **unconditional** moving, specify REPLACE=YES. This means that:

- With MOVE LIB=..., all sublibraries of the from-library are moved into the to-library. In the to-library, any existing sublibraries which have the same names as moved sublibraries are overwritten. Any existing sublibraries of the to-library which are not overwritten by moved sublibraries remain in the to-library.
- With MOVE SUBLIB=..., the to-sublibrary is first emptied, if it already exists, or created, if it does not yet exist. Then all members of the from-sublibrary are moved into the to-sublibrary;
- With MOVE mn.mt..., the specified member of the from-sublibrary is moved into the to-sublibrary. If a member of the specified name and type exists in the to-sublibrary, it is overwritten.

For **conditional** moving, specify REPLACE=NO or omit the REPLACE operand. This means that:

- With MOVE LIB=..., a sublibrary of the from-library is not moved if a sublibrary of the same name exists in the to-library. All other sublibraries of the from-library are moved, and all existing sublibraries of the to-library remain in that library;
- With MOVE SUBLIB=..., the from-sublibrary is not moved if the to-sublibrary already exists;
- With MOVE mn.mt..., the specified member is not moved if a member of the same name and type exists in the to-sublibrary.

When you specify members generically, each from-sublibrary member which matches the specification is compared singly with existing members in the to-sublibrary. Therefore, some of the specified members may be moved, while others are not.

## MOVE

### **LISt=Yes | No**

Specify YES to obtain a listing on SYSLST of the names and types of the members moved, together with the corresponding sublibraries.

This is especially useful if the members were specified generically or in the case of conditional moving, with REPLACE=NO.

If you specify NO, or omit the operand, no such listing is produced.

### **TLOCK=COPY | RESET | NORMAL**

Controls the locking status of the moved member in the target sublibrary. A locked member in the source library will not be deleted if it is locked.

#### **Target member does not exist:**

- TLOCK=COPY indicates that the target member gets the locking status of the source member.
- TLOCK=RESET or NORMAL indicates that the target member will be unlocked.

#### **Target member exists:**

If REPLACE=NO was specified, any library member or sublibrary that already exists in the to-library is not moved and therefore keeps its locking status, irrespective of the specified TLOCK operand.

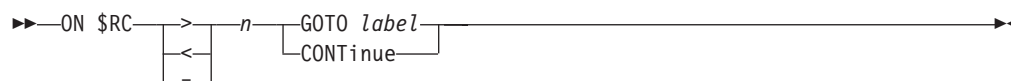
If REPLACE=YES was specified,

- TLOCK=COPY indicates that the locking status of the source member is to be copied to the target member, that is, the processed target member will be (un)locked if the source member was (un)locked. The source member will be deleted if it is unlocked. The target will be replaced even if it is locked.
- TLOCK=RESET indicates that the specified target member(s) will receive a lock status of 'unlocked', no matter whether the source member was locked or unlocked before moving. The source member will be deleted if it is unlocked.
- TLOCK=NORMAL indicates the following:
  - If a whole library is to be moved, the command will bypass those target sublibraries that contain locked members.
  - If a sublibrary is to be moved and the source or target sublibrary contains locked members, this sublibrary will not be moved.
  - If one or more members are to be moved and a target member already exists which is locked, that member is not moved (and a message is issued). The source members will not be deleted if they are locked.

Move will never delete locked members in the source library. Moving of a sublibrary that contains locked members will be bypassed. Moving of a locked member will be bypassed.



## ON (Set Global Condition)



The ON command allows conditional execution of librarian command streams in batch mode.

The Librarian program sets a return code each time processing of a command is completed. On successful completion, this return code is 0 or (for the COMPARE and TEST commands only) 2; in case of an error it is 4, 8 or 16, depending on the severity of the error. A return code of 16 is set only when a severe error has occurred, and the Librarian program has terminated abnormally. With a return code of 0, 2, 4 or 8, processing of the Librarian commands continues.

The ON command causes the Librarian program to test the return code after each following command. If the comparison of the return code with the specified number yields the value “true”, the specified action is taken.

For further details on the Librarian return codes and examples of conditional command execution, see “The Librarian Program” in the *z/VSE Guide to System Functions*.

### \$RC

Indicates the return code of any following Librarian command

### >|<|=

Indicates whether the specified action is to be taken when the return code is:  
 Greater than (>),  
 Less than (<) or  
 Equal to (=)

the specified number.

- n** Specifies the number with which the return codes are to be compared. It must be a whole number in the range 0..9999.

### GOTO label | CONTInue

Specify the action to be taken if the specified comparison yields the value “true”. The operand GOTO has the same effect as the GOTO command.

Processing continues with the command following the specified label. For “label”, substitute the name of a /. label command. This label must occur **after** the ON command.

If the label has been specified before the condition gets true, or no label has been specified, the librarian will read past the /& statement until the physical end-of-file.

If you specify CONTInue, processing continues with the command immediately after the command which caused the return code to be set.

## ON

### Notes:

1. The ON command is ignored if entered at SYSLOG.
2. There may be up to 30 active ON commands in one sequence of Librarian commands.
3. Each ON command remains active until it is overridden by another, or until end-of-file on SYSIPT.
4. /& will not be recognized as end-of-file when searching for a label.
5. A new ON command overrides all previous ON commands with the same condition, or with a condition which is included in the new one. For example:  
ON \$RC > 4 GOTO B

overrides:

```
ON $RC > 8 GOTO A
```

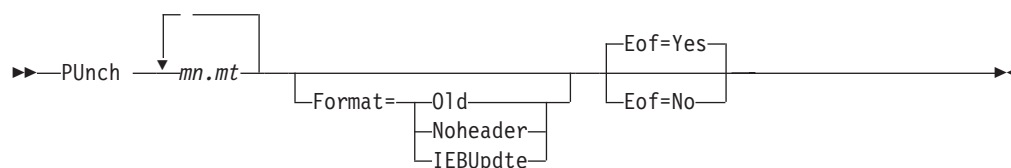
but does not override:

```
ON $RC = 4 CONT
```

6. If several ON commands are active, the condition of the most recent one is tested first.
7. The system default ON conditions are:  
ON \$RC>0 CONTINUE  
and  
ON \$RC=0 CONTINUE.

That is, any return code will allow processing to continue.

## PUNCH (Punch Member Contents)



The PUNCH command causes the contents of one or more members to be “punched” to the output device SYSPCH. Note that “punched” indicates the record format of the data produced, not to the type of storage medium used for output.

Members of type DUMP and members with a record length other than 80 (except phases), cannot be “punched”.

Unless you specify FORMAT=NOHEADER, the members are prepared for re-cataloging. That is, a CATALOG command with the operand REPLACE=YES, or a PHASE statement, is placed before each member. If applicable, EOD and DATA operands are added (see CATALOG command). Each member is followed by an EOD record.

The operand FORMAT=IEBUPDTE causes the PUNCH command to punch z/VSE library members in a format acceptable for MVS\*.

For phases, the PUNCH command will store the AMODE and RMODE attributes in the ESD card of the punched object. These AMODE and RMODE values represent the original specification defined by the Linkage Editor according to the ESD information of the linked object modules.

If the AMODE or RMODE was overridden by the Linkage Editor MODE control statement or the PARM field, a MODE control statement is additionally punched in front of the PHASE card.

Thereby, you can get the original AMODE and RMODE specification from the ESDs just by relinking the phase without the generated MODE control statement. The punched card decks without any modifications can always be used to relink the identical phase.

### **mn.mt**

Specifies the members to be punched. The sublibrary must be specified in a preceding ACCESS command.

### **Format=Old**

Allows members to be transferred from a current library to a library of a VSE-Version-1 format. If FORMAT=OLD is specified, the punched members will be prepared for cataloging by the MAINT program into a VSE-Version-1 library.

### **Notes:**

1. If this operand is specified, the naming conventions for members must conform to the rules for the old MAINT CATALx programs.
2. This operand is not applicable for members which have a user type. They are not prepared for re-cataloging.
3. This operand is ignored for members of the type PHASE. They are punched as usual.

## PUNCH

### **Format=Noheader**

Suppresses the punching of the CATALOG and EOD commands. Only the contents of the member are punched. This operand is ignored for members of type PHASE, and for phases which have been renamed to a user type.

### **Format=IEBUpdte**

Causes the PUNCH command to place in front of each member, instead of the CATALOG command, the following MVS™ control statement:

```
./ ADD NAME=membername
```

In addition, no EOD record is punched at the end of the members. If EOF=YES is specified, the following MVS control statement is written to SYSPCH after completion of the command:

```
./ ENDUP
```

### **Notes:**

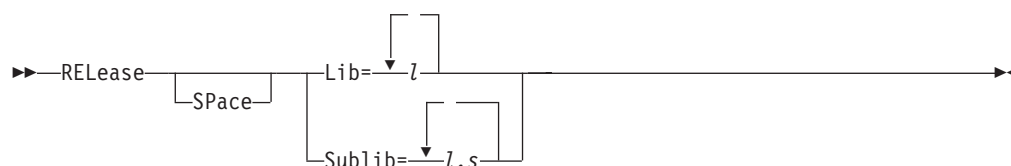
1. The FORMAT=IEBUPDTE operand is ignored for members of the type PHASE.
2. If FORMAT=IEBUPDTE has been specified, it must be included in all PUNCH commands of one LIBR run.
3. If FORMAT=IEBUPDTE is specified and SYSPCH is assigned to a tape device, only 80-byte records are punched, that is, no control characters are placed in front of the record.

### **Eof=Yes | No**

If EOF=YES is specified, or the operand is omitted, an end-of-file indicator (/\*) is written to SYSPCH after completion of the command.

If EOF=NO is specified, no end-of-file indicator is written. This makes it possible to collect members from different sublibraries in one punch file. Only the last member to be punched should have EOF=YES (or default).

## RELEASE (Release Unused Shared Space)



The RELEASE command overrides the library or sublibrary attribute AUTOMATIC. It is needed only when members have been deleted from a library or sublibrary which:

- Is shared by two or more VSE partitions; or
- Resides on a disk device shared by two or more processors.

In these cases, the space formerly occupied by deleted members is normally released only when the library or sublibrary is no longer shared. The RELEASE command causes this space to be released immediately.

Therefore, if several CPUs access the specified library or sublibrary, the RELEASE command must be given in all CPUs, to avoid inconsistency in results.

If you want to use the RELEASE command for the system sublibrary IJSYSRS.SYSLIB, first enter a SET SDL job control command at all CPUs sharing the sublibrary. The SET SDL command must be followed by the names of the phases that have been deleted (explicitly or implicitly), and that have entries in the SDL. This avoids SDL entries pointing to empty spaces.

### SPace

may be coded as a reminder to someone not familiar with your job stream that only free space, and not valid members, are affected by the command. Whether you omit or include the operand makes no difference to the function which is carried out.

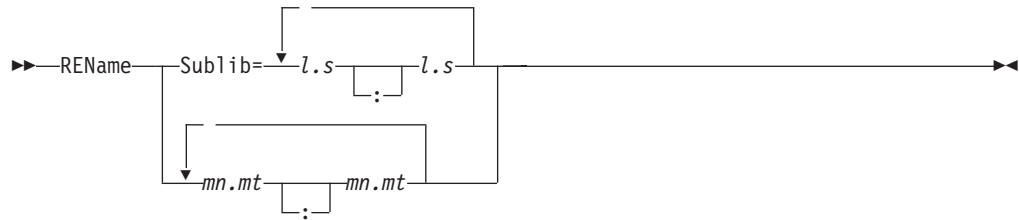
### Lib=l

Specifies in which library the space formerly occupied by deleted members is to be released.

### Sublib=l.s

Specifies in which sublibrary the space formerly occupied by deleted members is to be released.

## RENAME (Rename Sublibrary or Member)



The RENAME command changes the name and/or type of one or more members, or the names of one or more sublibraries. If the new name already exists, the function is not executed.

Sublibraries which contain **locked** members will not be renamed. Likewise, renaming of all locked members is ignored.

### Sublib=l.s

Specifies the old (first operand) and new (second operand) names of the sublibrary to be renamed. The library name in the second operand must be the same as in the first, and can be specified by an equals sign. The system sublibrary IJSYSRS.SYSLIB cannot be renamed. A sublibrary can only be renamed if it is not in use in another VSE partition. If a sublibrary was specified in a LIBDEF statement, the LIBDEF statement must be dropped or changed accordingly before the RENAME command is entered. Any ACCESS or CONNECT commands for the renamed sublibraries are dropped.

The RENAME S=... command cannot be processed when the specified sublibrary is in use. If the sublibrary to be renamed is on a disk device shared by two or more CPUs, the system issues a message prompting the operator to verify that renaming is to take place.

### mn.mt

Specifies the old (first operand) and new (second operand) names and types of the members to be renamed. The sublibrary must be specified in a preceding ACCESS command.

Remember that the member names and types must be 1..8 characters long.

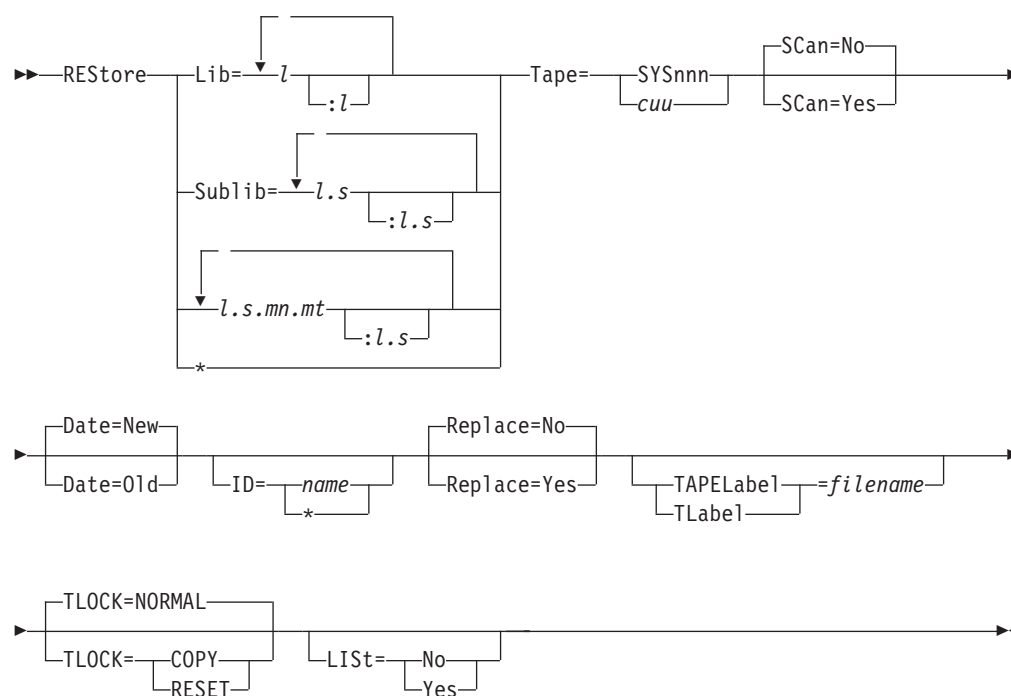
If the new name or type are specified generically, the corresponding part of the "old" operand must also be specified generically. If the name or type is not to be changed, you may specify an equals sign for it in the second operand.

Note that renaming to a new type must not imply a change of internal representation. Members of the types PHASE and DUMP have a different internal representation from other types. Renaming into a user type is always possible, but you must be careful not to violate the above restriction when renaming into a predefined type.

If a member of the specified new name and type existed in the sublibrary before the command was carried out, its "put-in-sublibrary" time stamp is left unchanged, and the "last-replaced" time stamp is updated.

If such a member **did not** exist in the sublibrary, the "put-in-sublibrary" time stamp is set, and the "last-replaced" time stamp is empty.

## RESTORE (Restore Backed-up Library, Sublibrary or Member)



The RESTORE command causes the libraries, sublibraries, members or SYSRES files which were backed up using the BACKUP command to be restored to disk. The disk address and extents of the libraries are determined from DLBL, EXTENT and ASSGN information in the label area. If no label information is available for a library, it will not be restored. If several libraries, sublibraries, SYSRES files or members are to be restored, the sequence of the restoring is determined by the sequence as stored on the backup file. The RESTORE function does not reposition the backup tape after reading the specified backup files.

You may also restore members selectively out of backed-up sublibraries. Members with the same names as the restored members will be overwritten in the to-sublibrary or not, depending on the REPLACE operand of the RESTORE command.

Libraries and sublibraries keep the same attributes (for example, REUSE, MSHP) after restore as they had before backup. There is one exception to this rule. If a sublibrary is restored into a target sublibrary which already exists, the REUSE attribute of the **target** sublibrary is kept.

A library may not be restored if it is in use.

### Lib=l

Specifies a library to be restored. Restoring of a library or SYSRES file includes creating it. The location of the libraries to be restored must be outside the extent of the IPLed system. SYSRES files to be restored must not be on the same disk as the IPLed system. The number of extents for a library may change during BACKUP/RESTORE, the disk type may be different, and a library which resided in VSAM managed space before may be restored to

## RESTORE

non-VSAM space and vice versa. If a SYSRES file with the name IJSYSRS is to be restored online, it must be given a new name, which must be of the form IJSYSR1...IJSYSR9.

- :l Specifies a new name for the library to be restored, and, if specified, this new name is used by the librarian to determine the label and extent information to be used in restoring the library. This label and extent information must be given before the RESTORE command, using DLBL and EXTENT job control statements. The rules for coding these statements are the same as those given in the description of the librarian DEFINE command. SYSRES files can be restored as private libraries by giving them new names other than IJSYSRn, but private libraries must remain private. The specification of the DEFINE EXTENTS parameter of the library on the backup tape is used for the library to be restored.

### Sublib=l.s

Specifies a sublibrary to be restored. A sublibrary can be restored only into an existing library. Restoring a sublibrary includes creating it, if a sublibrary with the same name does not already exist in the target library. If a sublibrary with the same name does exist in the target library, this will be overwritten or not, depending on the REPLACE operand of the RESTORE command. The system sublibrary of the IPLed system (IJSYSRS.SYSLIB) cannot be deleted, so a backed up version of it must be given a new name when it is restored. The specified sublibrary may be part of a library which was backed up on tape as a whole. It is not necessary to restore the whole library for the sake of a few sublibraries.

### :l.s

Specifies a new target sublibrary. By specifying this operand you can cause the sublibrary to be restored to a library other than that from which it was backed up, and you can give it a new name. If the library or sublibrary is the same as in the first operand, you may specify an equals sign for it. The REUSE attribute of the target sublibrary is retained, if this sublibrary existed. Otherwise, the attribute of the sublibrary from the tape is used.

### l.s.mn.mt

Specifies the member(s) to be restored, and in which backed-up sublibrary they are to be searched for. The target sublibrary on disk must be specified in a preceding ACCESS command, or by :l.s.

This operand is positional, that is, it must be coded as the first operand after the command name.

If a member of the specified name and type existed in the target sublibrary before the command was carried out, its **CREATION DATE** time stamp is left unchanged, and the **LAST UPDATE** time stamp is updated.

If such a member **did not** exist in the target sublibrary, the **CREATION DATE** time stamp is set, and the **LAST UPDATE** time stamp is empty.

### :l.s

The specified members will be copied into this sublibrary, dependent on the REPLACE operand of the RESTORE command. They may be restored to the system sublibrary (IJSYSRS.SYSLIB) of the IPLed system. The specified member(s) may be contained in libraries or sublibraries which were backed up on tape as a whole.

**Note:** When replacing existing members, be sure to provide enough space in the target library for both the old and new members. This is necessary because the old members are deleted only after restore of the new ones.



- \* Specifies that all libraries, sublibraries, or members on the backup file are to be restored. If the backup tape contains only members, the target sublibrary on disk must be specified in a preceding ACCESS command.

This operand is positional, that is, it must be coded as the first operand after the command name. The "\*" operand may also be specified for a backup file containing libraries of pre-VSE/SP Version 2 format if the operand SCAN=YES is specified with it.

#### **Tape=SYSnnn | cuu**

Specifies the logical unit or device address of the tape containing the backup file. When using a multi-volume backup tape, specify the alternate tape(s) using the job control statement ASSGN.

If the ID=name operand is not used, the first RESTORE command specifying a newly mounted tape processes the first backup file on the tape. The tape is not repositioned. Any following RESTORE command (without an ID=name specification) begins processing at the start of the next backup file on the tape. This is true even if the first RESTORE does not process all of the first file.

When the last backup file on the tape has been processed, or the name specified in the ID operand has not been found, the tape is positioned at the tape mark before the End-of-Backup record. (For details, see the description of the librarian command BACKUP).

#### **Date=New | Old**

DATE=NEW indicates that when you restore a library or sublibrary, the creation dates in their time stamps are set to the time of the restore. Likewise, all members in the library or sublibrary receive new time stamps: the **CREATION DATE** field shows the date and time of the restore.

DATE=OLD indicates that the date and time stored on the backup tape is to be used.

If you specify DATE=OLD for a member restore, the **CREATION DATE** or the **LAST UPDATE** time stamp might be older than the creation date of the sublibrary or library. If you specify DATE=OLD for a sublibrary restore, the restored sublibrary may have an earlier creation date than the containing library.

DATE=OLD cannot be used for:

- A stand-alone restore run.

You can use this option for any backup that was taken by the librarian of VSE/SP Version 2. However, the **LAST UPDATE** time stamp is not retained if the backup tape was created under VSE/SP 3.1.2 or earlier. The **CREATION DATE** time stamp is retained.

The DATE operand is ignored if SCAN=YES is also specified on the RESTORE statement.

#### **LISt=Yes | No**

If you specify LIST=YES, the names of the restored libraries, sublibraries and (if members were specified in the first operand) members will be printed on SYSLST. If LIST=NO is specified no listing will be produced. If libraries or sublibraries are specified in the first operand, the default is LIST=YES; if members are specified, the default is LIST=NO.

#### **SCan=Yes | No**

If you specify SCAN=YES, the restore function is not performed. Instead, information about the data on the backup file is printed on SYSLOG, if the

## RESTORE

command was entered from there, otherwise on SYSLST. The kind of information provided depends on what you have specified in the first operand, as follows:

If \* is specified in the first operand, the names of all libraries, sublibraries, or members found on the backup file are printed. If specific library, sublibrary or member names were used in the first operand, their names are printed, and a message informs you whether or not they are present on the backup file.

The information on libraries and sublibraries also contains the space requirements for all supported disk devices. This enables you to provide appropriate DLBL and EXTENT information if required.

If you specified generic member names in the first operand, all member names which match the generic specification will be printed.

SCAN=YES can be used for private libraries in pre-VSE/SP Version 2 format on backup tapes. In this case, an estimate of the space required on all supported disk devices is given.

### **ID=name | \***

Specifies the name of the backup file to be searched for. The name must have been specified in the BACKUP command which created the file. It may consist of 1 to 16 characters enclosed in quotes. If only alphanumeric characters are used, the quotes may be omitted.

For an unlabeled tape, the backup tape is searched from the current tape position until the specified file is found, or the last backup file has been reached, that is, until an End-of-Backup record is encountered. For a labeled tape, the tape is not searched. The name of the current backup file is checked against the specified name.

If you specify ID=\*, the entire backup tape is searched from the current position on.

The ID operand is not applicable if libraries from pre-VSE/SP Version 2 backup tapes are to be restored.

### **Replace=Yes | No**

Controls the restoring of backed-up libraries and sublibraries and the merging of pre-VSE/SP Version 2 libraries and members into existing ones. If REPLACE=NO is specified, or the operand is omitted, the entities of the backed-up library or sublibrary are not restored if an entity with the same name already exists in the target entity. With REPLACE=YES, the entire backed-up library or sublibrary, or all specified members, are restored regardless of whether there are duplicate names.

**Note:** VSE/SP is the predecessor system of VSE/ESA.

### **TAPLabel | TLabel=filename**

Indicates that the involved library object is to be restored from a tape with standard labels. **filename** is the seven-character file name that you specified in the TLBL statement for your backup file.

### **TLOCK=COPY | RESET | NORMAL**

Controls the locking status of the restored member in the target sublibrary.

#### **Target member does not exist:**

- TLOCK=COPY indicates that the target member gets the locking status of the member on the backup tape.

- TLOCK=RESET or NORMAL indicates that the target member will be unlocked.

**Note:** During backup, the locking status of a member is always copied to the backup tape.

**Target member exists:**

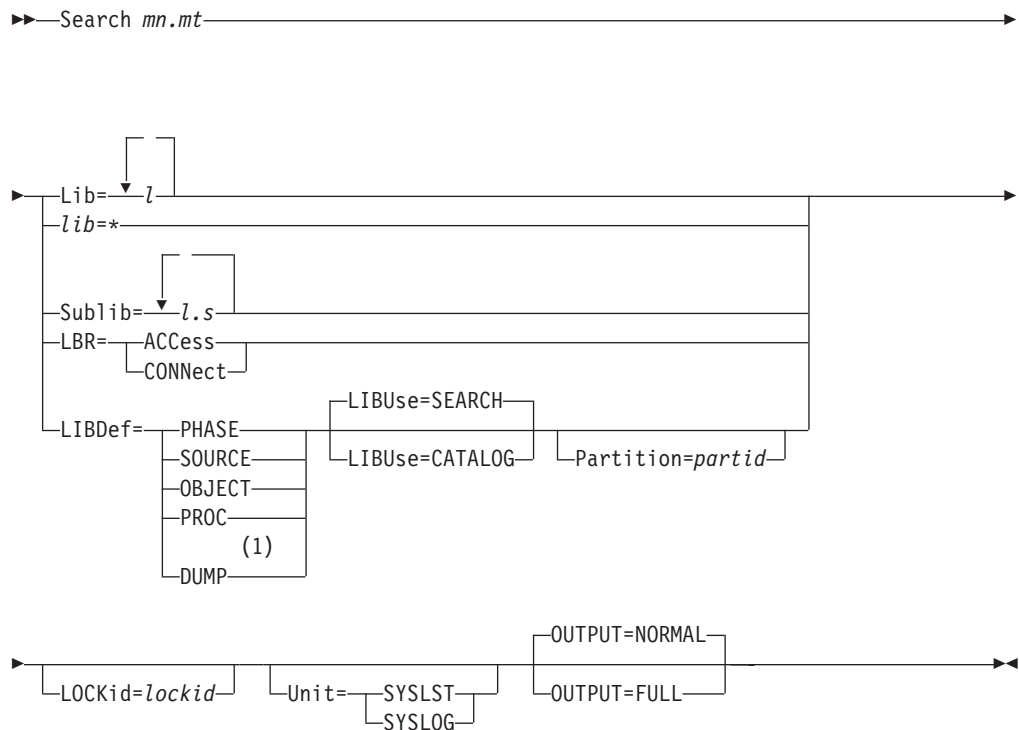
If REPLACE=NO was specified, any library member or sublibrary that already exists in the to-library is not restored and therefore keeps its locking status, irrespective of the specified TLOCK operand.

If REPLACE=YES was specified,

- TLOCK=COPY indicates that the locking status of the member on the backup tape is to be copied to the target member, that is, the processed target member will be (un)locked if the source member was (un)locked. A locked member will be replaced.
- TLOCK=RESET indicates that the specified target member(s) will receive a lock status of 'unlocked', no matter whether the locking status of the member on the backup tape was locked or unlocked. The member will be restored in any case and the target will be replaced.
- TLOCK=NORMAL indicates the following:
  - If a whole library is to be restored and any target sublibrary contains locked members, the library will not be restored.
  - If a sublibrary is to be restored and the target sublibrary contains locked members, this sublibrary will not be restored.
  - If one or more members are to be restored and a target member already exists which is locked, that member is not restored (and a message is issued).

The stand-alone version of this function restores a single SYSRES file. If the backup file contains more than one SYSRES file, one can be selected. Any private libraries on the same backup file can only be restored online. For information on how to run the stand-alone version of the RESTORE function, see "Restoring a SYSRES File Stand-Alone" in the *z/VSE Guide to System Functions*.

## SEARCH (Search for Library Member)

**Notes:**

- 1 For LIBDEF=DUMP, LIBUSE=CATALOG is default.

The SEARCH command is used to search for a library member in the specified library or libraries. If the member was found, the library (or list of libraries) where the member was found is printed. If the member was not found, a return code of 2 is set.

**mn.mt**

Specifies the member name and member type of the member to be searched. The specification may be generic.

**Lib=l**

Specifies the library or libraries in which the member is to be searched.

**Lib=\***

Specifies that the member will be searched in all libraries which are currently open in the system.

**Sublib=l.s**

Specifies the sublibrary or sublibraries in which the member is to be searched.

**LBR=ACCess | CONNect**

Indicates that the member is to be searched in the chains which are created by the ACCESS or CONNECT command.

**LIBDef=PHASE | SOURCE | OBJECT | PROC | DUMP**

Indicates that the member is to be searched in the active LIBDEF chain of the specified type. (The LIBDEF chain is defined in the job control LIBDEF command). The LIBDEF chain can be further identified with the LIBUSE and the PARTITION operands.

**LIBUse=SEARCH | CATALOG**

Specifies whether the SEARCH or the CATALOG library list of the job control LIBDEF command is to be searched:

For LIBDEF=PHASE, both LIBUSE=SEARCH (which is the default in this case) and LIBUSE=CATALOG are valid.

For LIBDEF={SOURCE | OBJECT | PROC}, only LIBUSE=SEARCH is valid, which is also the default in this case.

For LIBDEF=DUMP, only LIBUSE=CATALOG is valid, which is also the default in this case.

**Partition=partid**

Indicates the partition in which the specified LIBDEF chain lies. The default is the partition in which the SEARCH command was entered.

**LOCKid=lockid**

Indicates that only the members locked with the specified **lockid** are to be searched for in the specified libraries, sublibraries or chains. The **lockid** is a string of up to eight alphanumeric characters; it can also be generic.

**Unit=SYSLST | SYSLOG**

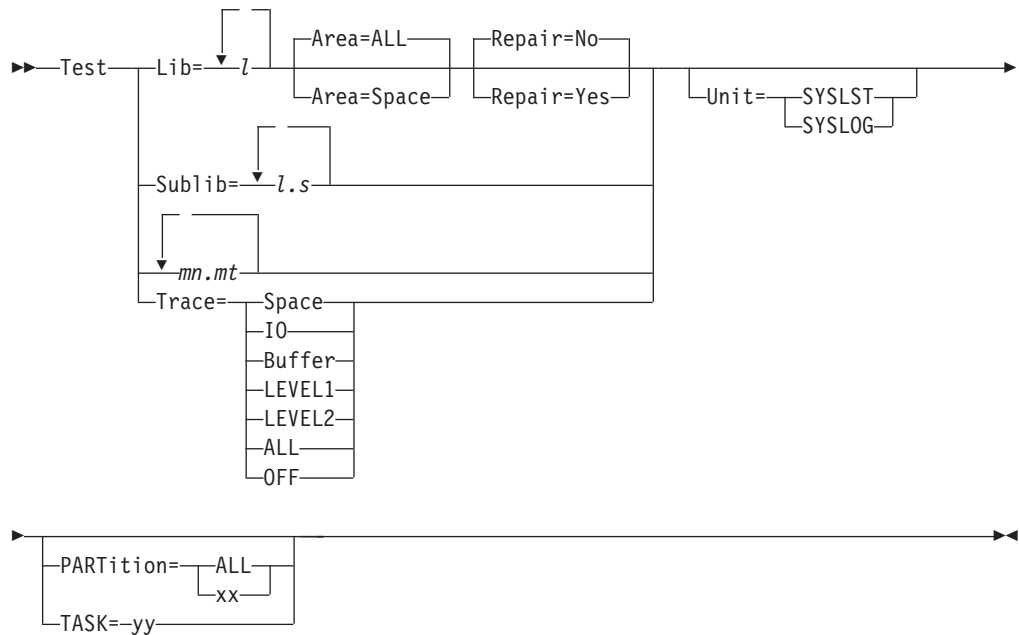
Specifies the output device where the result of the SEARCH command is to be printed. If the SEARCH command is issued from SYSLOG, the default output device is also SYSLOG. If the command is issued from SYSIPT, the default output device is SYSLST.

**OUTPUT=NORMAL | FULL**

Controls the type of information provided. OUTPUT=FULL provides additional information about the affected sublibraries such as creation date, number of sublibraries, device and volume information.

## TEST

### TEST (Test Library Integrity)



The TEST command should be used only on request of IBM service personnel. It checks the structure and contents of a library, sublibrary or member for consistency and correctness, and to provide a trace function for librarian services at different levels.

If TEST detects any inconsistency or incorrectness in a library, sublibrary or member, the librarian sets a return code of 2.

When a library is tested, free library blocks are tested as well.

To use the TEST command on a resource protected by the Access Control Function, you need READ access to the affected library.

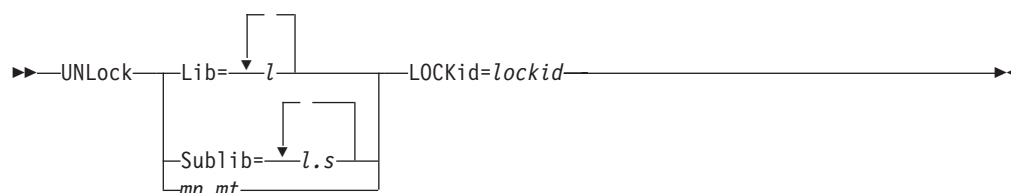
To ascertain a possible library problem, follow this procedure:

1. Run TEST LIB=l for the library suspected of causing the problem.
2. If the TEST output does not show error lines, the problem was not caused by this library.

If the TEST output shows errors, then:

3. Run BACKUP and RESTORE for the library.
4. Run TEST LIB=l again for the same library.
5. If the TEST output does not show error lines, the problem is probably solved.  
If the TEST output does show errors again, then:
6. Contact your support center. There is probably a system error.

## UNLOCK (Unlock Member)



The UNLOCK command causes a library member that has been locked for any write or update access to be unlocked again. The member will be unlocked only if the specified **lockid** corresponds with the **lockid** with which this member was locked.

The UNLOCK command unlocks library members unless:

- You have no UPDATE access right for the specified library, sublibrary, or member,
- The member is locked with a lockid that does not match the one specified in the LOCK command,
- The member is not locked.

### **mn.mt**

Specifies the member name and member type of the member to be unlocked. Generic specification is allowed. The sublibrary of the member must be specified in a previous ACCESS command.

### **Lib=l**

Specifies the library or libraries containing the member to be unlocked.

### **Sublib=l.s**

Specifies the sublibrary or sublibraries containing the member to be unlocked.

### **LOCKid=lockid**

Specifies the lock identifier with which the member was locked (with the corresponding LOCK command). The **lockid** is a string of up to eight alphanumeric characters; it can also be generic.

For example:

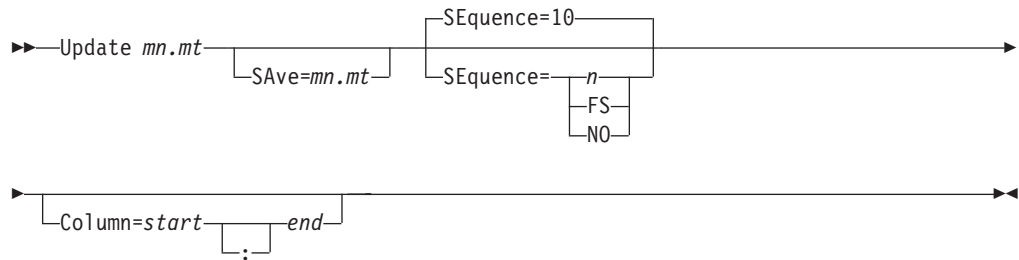
```
ACCESS S=TEST.S1
UNL ALF.A LOCK=BOB
```

will unlock member **ALF.A** in sublibrary **TEST.S1** if this member is locked with lockid **BOB**. Otherwise an error message is issued.

A member will be unlocked only if the specified **lockid** corresponds with the **lockid** with which this member was locked.

If UNLOCK for an explicitly specified (non-generic) member failed, the command leaves the member unchanged.

## UPDATE (Alter Member Contents)



The UPDATE command allows you to modify the contents of a member by adding, deleting or replacing lines. If you wish, you can save the unmodified version under a new name or type or both. The command applies to all member types which can be created by the CATALOG command.

### **mn.mt**

Specifies the member to be updated. The sublibrary must be specified in a preceding ACCESS command. If the member is locked (via the LOCK command), the UPDATE function will not be carried out.

### **SAve=mn.mt**

Specifies that the unmodified version of the member is to be saved under the name and type specified. If a member with the same name and type already exists in the sublibrary, the UPDATE function will not be carried out. If a new type is specified, it must not imply a change in the internal representation of the member.

### **SEquence=10 | n | FS | NO**

Controls the resequencing of the member being updated. FS may be specified only if the sequence field in the member is numeric without leading blanks. If you omit the operand, SEQUENCE=10 will be assumed by default.

**n** must be a decimal number from 1 to 999, and specifies the increment between line numbers which will be used for resequencing. The first line will be given the value n.

**FS** specifies fixed sequence; the current line numbers will not be changed. The updates are checked to ensure that a valid sequence is retained.

**NO** specifies that the order of the records in the member will not be checked. The updates must be supplied in ascending order. The sequence number may consist of any alphanumeric characters. If it is shorter than the length specified in the COLUMN operand, it must be padded on the right with blank characters. Sequencing is not checked.



**Column=start:end**

Specifies the start and end of the sequence field in the member. This may be located anywhere within the line, and can be 1 to 8 characters long. The following defaults will apply if the operand is omitted:

if SEQUENCE=n

or SEQUENCE=NO: COLUMN=77:80

if SEQUENCE=FS: COLUMN=73:78

## UPDATE Subcommands

The UPDATE subcommands )ADD, )DEL and )REP have as their operand the sequence number(s) of the line(s) of the member to which the subcommand applies. If the member you wish to update has no sequence numbering, you can cause the librarian to generate it by issuing the following command sequence:

```
EXEC LIBR          call the librarian program
ACCESS S=lib.sublib  access appropriate sublibrary
UPDATE membername.membrty  "update" the member without input
)END              end update, causing renumbering
```

These commands cause the lines of the specified member to be sequence numbered in columns 77 to 80 using an increment of 10, starting at 10. These are the default values used by the librarian. Should you wish to use a different increment or a different length or position for your sequence numbering, specify the appropriate values in the SEQUENCE and COLUMNS operands of the UPDATE command.

)DEL and )REP act on the specified line(s), )ADD inserts the provided data **after** the specified line. Specification is by means of the sequence number.

The updates applied with one UPDATE command must be in ascending order. That is, the first operand of an )ADD, )DEL or )REP subcommand must be **greater than** the first operand of any preceding )ADD, )DEL or )REP subcommand.

The length and position within the line of the field containing this sequence number (the sequence field) must be specified in the COLUMN operand of the UPDATE command, and the length of the seq-no operands of the subcommands must not exceed this specification.

When using SEQUENCE=n or FS on the UPDATE command, you may omit leading zeros in the subcommand operands. With SEQUENCE=NO, the sequence number or character string specified is padded on the right with blank characters if necessary.

If you are adding or replacing lines in a member using SEQUENCE=NO, the input lines following the )ADD or )REP subcommand must contain the sequence number or character string.

When using )ADD or )REP with SEQUENCE=n, you need not provide sequence numbering in the input lines, and if you do so, you may omit leading zeros. The member in which you want add or replace lines must, of course, be sequence numbered.

**Note:** Update subcommands must always start with a ) in column 1, with or without one blank character in column 2, for example:

```
)ADD ...
or
) ADD ...
```

## )ADD (Add Line to Member)

The )ADD subcommand indicates that the lines following it are to be added to the member specified in the UPDATE command.

►►)ADD *seq\_no*◄◄

### **seq-no**

Represents the sequence number of the line in the member after which the new lines are to be added. To add new lines in front of the first line of the member, code 0 for seq-no. Adding lines in front of the first line is not possible if you have specified SEQUENCE=NO in the UPDATE command.

If you specify sequence numbers in the input lines following this subcommand, be sure that their position and length correspond with the COLUMN operand in the UPDATE command.

## UPDATE )DEL

### )DEL (Delete Line from Member)

The )DEL subcommand causes the deletion of lines from the member specified in the UPDATE command.

```
▶▶—)DEL first_seq_no , last_seq_no , *▶▶
```

#### **first-seq-no,last-seq-no**

Represent the sequence numbers of the first and last lines of a section to be deleted. If **last-seq-no** is not specified, the line represented by **first-seq-no** is the only line deleted. To delete all lines from the line specified in first-seq-no to the end of the member, specify \* in place of last-seq-no.

**)END (Finish Update)**

The subcommand )END has no operand. Issue it to inform the system that input for the required UPDATE function is complete.

▶—/\*—▶

## UPDATE )REP

### )REP (Replace Line in Member)

The )REP subcommand causes the replacements of lines from the member specified in the UPDATE command.

```
▶▶—)REP first_seq_no , last_seq_no
, *
```

#### **first-seq-no,last-seq-no**

Represent the sequence numbers of the first and last lines of a section to be replaced. The **first-seq-no** must not be zero. Any number of new lines can be added to a member when a section is replaced. The number of lines added need not equal the number of lines being replaced. To replace all lines up to the end of the member, specify \* in place of last-seq-no.

If you specify sequence numbers in the following input lines, be sure that their position and length correspond with the COLUMN operand on the UPDATE command.

## /. (Label Statement)

▶▶—/. *label*—————▶▶

The /. label statement is used in conditional command streams. It marks a point in the command stream up to which commands can be skipped using a GOTO command or the GOTO action of an ON command. The Librarian label statement corresponds to the job control label statement.

/. Indicates that this is a label. These characters must be in positions 1 and 2 of the command followed by at least one blank character.

### **label**

Specifies the name of the label. This must be 1 to 8 alphanumeric characters.

You must use this name in the label operand of the GOTO command which addresses the label.

**Note:** The /. statement is ignored if entered from SYSLOG.

## End-of-Data

### /+ (End-of-DATA)

The End-of-Data statement for input to the librarian CATALOG command is /+. This is used for data of all types, whether procedures, source code or other user data.



Column 1 contains a slash (/) and column 2 a plus sign (+). Column 3 must be blank.

The /+ statement is also used by job control as an End-of-Procedure statement. If this is the last statement of a member to be cataloged, the librarian recognizes the end of the input data and includes an End-of-Procedure mark at the end of the cataloged member.

If, however, a /+ statement must be included as part of a member to be cataloged, the CATALOG step for this member must have an End-of-Data statement other than /+. You can define the alternative End-of-Data statement in the EOD operand of the librarian CATALOG command. The need for an alternative EOD statement arises, for example, when you catalog a procedure which itself contains librarian CATALOG commands.



## **/\* or END (Librarian End-of-Session)**

These statements indicate to the librarian program that no more librarian commands follow.

### **Format for SYSIPT**

▶▶—/\*—————▶▶

### **Format for SYSLOG**

▶▶—END—————▶▶

The SYSIPT format is used when a librarian job stream for batch execution is being prepared. The SYSLOG format is used to end an interactive librarian session at the system console.

When the librarian program receives either form of the End-of-Session command, it gives control of the partition to the job control program. The highest return code set during the librarian session or step is passed to job control. You can test this return code using the job control statements IF and ON.



---

## Chapter 6. Edited Macro Service Program (ESERV)

The ESERV program de-edits assembler macros of type E created by the DOS/VSE assembler, and punches and/or displays the macros in source format. It is also possible to update the source form of the macro before output. For further information on the use of ESERV, see "Processing Macros with the ESERV Program" in the *z/VSE Guide to System Functions*.

To run ESERV, the following job control statements are necessary:

```
// ASSGN SYSPCH,uuu
// LIBDEF SOURCE,SEARCH=lib.sublib
// EXEC ESERV
```

(If an appropriate ASSGN and LIBDEF are already valid for the partition in which ESERV is to run, these statements need not be included.)

One of the following control statements is required: DSPLY, PUNCH or DSPCH.

To verify the macro in question, or to update it before it is displayed or punched, one or more of the following statements may then be entered:

) **ADD**

to add statements at a specified point in the macro;

) **COL**

to define the location and length of the sequence number field;

) **DEL**

to delete specific statement(s);

) **REP**

to replace specified statement(s);

) **RST**

to indicate that sequence numbering starts at a lower number than that of the specified preceding statement within a macro;

) **VER**

to verify the contents of a specified statement.

) **END**

to indicate the end of update statements. This statement **must** be used if any of the above verify or update statements are used.

Column 1 of these update statements must contain a right parenthesis, and there must be one blank before and at least one blank after the operation code.

If the update commands are entered after a DSPLY, PUNCH or DSPCH statement which specifies several members, they are applied to the last-named member.

/\* must be entered after the last ESERV control statement to indicate end-of-input on SYSIPT.

The syntax of all ESERV control statements is described in the following section.

---

## ESERV Control Statements

### GENCATALS (Specify Macro Output Format)

GENCATALS must follow the EXEC ESERV statement directly.

This statement must start in or after **column 2**.

▶▶—GENCATALS—————▶▶

This causes a librarian catalog statement for a member “bookname.A” (or “bookname.D”, if the OPTION SUBLIB=DF is in effect on the system) to be placed before each macro, and a /\* to be placed after each macro. This allows the SYSPCH output to be used as SYSIPT for the librarian program to catalog the de-edited macro with the appropriate member type.

## DSPLY, PUNCH, DSPCH (Specify Output Destination)

This statement must follow the GENCATALS statement. It can act on one or more edited macros in one ESERV run.

The statement must start in or after **column 2**.



### DSPLY

De-edits macros and displays them on SYSLST.

### PUNCH

De-edits macros and punches them on SYSPCH.

### DSPCH

De-edits macros, punches them on SYSPCH and displays them on SYSLST.

### **type.bookname**

Specifies the member name and member type of the macro to be de-edited. The sublibrary in which this member is to be searched for must be specified previously in a LIBDEF job control statement.

Verify and update control statements may follow these statements. If several macros were specified, the verify and update functions are applied to the **last** one specified.

## ) ADD (Add Statement to Macro)

The ) ADD statement indicates that the source statement(s) following it are to be inserted in the macro, and specifies at what position.

►► ) ADDseq\_no ─┬─  
                  └+rel┘

### seq-no

Indicates the sequence number of the macro definition statement **after** which the new source statements are to be inserted. The sequence number is 1..8 decimal digits, as specified in the ) COL statement.

### +rel

Indicates the position of the macro statement after which the new statements are to be added, **relative** to the statement specified in "seq-no".

**seq-no[+rel]** must be greater than the **(last-)seq-no[+rel]** of any preceding update control statement. An ) ADD statement may, however, reference the same statement as an immediately preceding ) VER statement.

## ) COL (Control Macro Statement Numbering)

The ) COL statement specifies the position of the sequence number within the source statements of the de-edited macro. If it is used, this statement must immediately follow the DSPLY, PUNCH or DSPCH statement to which it applies.

►—) COL *startcol,n*—————►

### **startcol**

Specifies the column in which the sequence number is to start. It must be a decimal integer; the valid range is 73..80. The default value is 73.

**n** Specifies the length of the sequence number. It must be a decimal integer; the valid range is 1..8. The default value is 6.

## ) DEL (Delete Statement from Macro)

The ) DEL statement causes deletion of one or more source statements from the de-edited macro.

►► ) DEL *first\_seq\_no* +rel , *last\_seq\_no* +rel ◄◄

### **first-seq-no**

Specifies the sequence number of the **first or only** source statement to be deleted from the de-edited macro. The sequence number is 1..8 decimal digits, as specified in the ) COL statement.

### **last-seq-no**

Specifies the sequence number of the **last** of a series of source statements to be deleted from the de-edited macro. The sequence number is 1..8 decimal digits, as specified in the ) COL statement.

### **+rel**

Indicates the position of the first or last statements to be deleted, **relative** to the statement with the specified sequence number. **rel** must be a decimal integer, and may be 1..4 digits long.

**first-seq-no[+rel]** must be greater than the **(last-)seq-no[+rel]** of any preceding update control statement. A ) DEL statement may, however, reference the same statement as an immediately preceding ) VER statement.



## ) END (Finish Macro Update)

The ) END statement indicates the end of ESERV update or verify statements on SYSIPT. It is required in every update run.

▶—) END—▶

## ) REP (Replace Statement in Macro)

The ) REP statement indicates that the following source statements on SYSIPT are to replace one or more existing statements in the de-edited macro.

►► ) REP *first\_seq\_no* +rel , *last\_seq\_no* +rel ►►

### **first-seq-no**

Specifies the sequence number of the **first or only** source statement to be replaced in the de-edited macro. The sequence number is 1..8 decimal digits, as specified in the ) COL statement.

### **last-seq-no**

Specifies the sequence number of the **last** of a series of source statements to be replaced in the de-edited macro. The sequence number is 1..8 decimal digits, as specified in the ) COL statement.

### **+rel**

Indicates the position of the first or last statements to be replaced, **relative** to the statement with the specified sequence number. **rel** must be a decimal integer, and may be 1..4 digits long.

**first-seq-no[+rel]** must be greater than the **(last-)seq-no[+rel]** of any preceding update control statement. A ) REP statement may, however, reference the same statement as an immediately preceding ) VER statement.

## ) RST (Change Macro Statement Numbering)

The ) RST statement causes the sequence numbers of the statements in a macro definition to restart at a lower number after the statement specified in the ) RST operand.

►► ) RST *seq\_no* +rel ◄◄

### seq-no

Specifies the sequence number of the source statement after which the new series of sequence numbers starts.

### +rel

Indicates the position of the statement after which the new series of numbers is to start, **relative** to the statement with the specified sequence number. **rel** must be a decimal integer, and may be 1..4 digits long.

If an ) ADD, ) DEL or ) REP operation is performed on the **last** statement in a sequence number series, the ) RST statement must reference the first statement **after** the statement specified in the ADD, DEL or REP.

## ) VER (Verify Contents of Macro Statement)

The ) VER statement causes all or part of the specified source statement in the de-edited macro to be verified against the contents of the statement following ) VER statement on SYSIPT.

The first string of characters, of the length specified in the **len** operand, are compared. If the strings do not match, an error message is issued.

►►) VER *seq\_no* +rel, *len* ◀◀

### seq-no

Specifies the sequence number of the source statement to be verified in the de-edited macro. It must be a decimal integer 1..8 digits long, as specified in the ) COL statement.

### +rel

Indicates the position of the source statement to verified, relative to the statement specified in "seq-no". It must be a decimal integer, 1..4 digits long. **seq-no[+rel]** must be greater than the **(last-)seq-no[+rel]** of any preceding update statement.

### len

Specifies the length of the field to be verified. It must be a decimal integer; the valid range is 1..80.

---

## Chapter 7. System Buffer Load (SYSBUFLD)

SYSBUFLD is a service program which loads UCBs (universal character set buffers) and FCBs (forms control buffers) of VSE supported line printers, except 3800. The buffer image phases for the UCB load operation must reside in a sublibrary; for the FCB load operation, the phases may reside in a sublibrary or the information may be read from SYSIPT (following the FCB control statement). For information on how to use SYSBUFLD under VSE/POWER, refer to *VSE/POWER Administration and Operation*.

SYSBUFLD is executed in your job stream whenever it is necessary to change the contents of the UCB and/or FCB of a specific printer. Execution is initiated with the statement:

```
// EXEC SYSBUFLD
```

When the access control function is active (SEC=YES was specified in the IPL command SYS), note that:

- The UCB and FCB phase names you use must start with '\$\$B';
- The phases must be cataloged in a protected library;
- You must establish access to this.

---

### Control Statements

Once started, SYSBUFLD reads, from SYSIPT, control statements which identify the printer and specify the buffer image to be loaded. The last statement is followed by a /\* statement. The control statements are BANDID, FCB and UCB.

#### BANDID

The BANDID control statement can be used only for 4248 printers. Use it to ensure that the correct band for the output of the following job is mounted.

```
▶▶—BANDID SYSxxx, band_id ,FOLD ,NOCHK▶▶
```

#### SYSxxx

Is the logical unit assigned to the 4248 printer for which the SYSBUFLD run is being performed. For SYSxxx, specify SYSLST or the programmer logical unit assigned to the printer.

#### band-id

Specifies the identifier of the required print band.

If you omit the operand, no print-band verification takes place. This is meaningful only if you want to change the output characteristics to FOLD or NOCHK.

If you specify a band identifier, a console message tells the operator which band is needed, if this band is not already mounted. The identifier of the band needed is repeated on the printer panel.

#### FOLD

Causes lowercase characters to be printed as uppercase characters.

**NOCHK**

Causes a data check to be suppressed if it results from a mismatch between a print character and the band-image buffer.

**FCB**

The FCB control statement is used to load the FCB of a printer.



**SYSxxx**

Identifies the printer whose FCB is to be loaded. The printer must be a 4248, 5203, or PRT1 device; SYSxxx must be SYSLST or a programmer logical unit; SYSxxx may be SYSLOG if, for any reason, SYSLOG has to be assigned to a line printer.

**phasename**

Specifies the name of the phase which contains the required buffer image. If the phase name is omitted, an FCB image from SYSIPT is assumed.

**NULMSG**

Indicates that the 80-character verification message, which follows the buffer image in the specified phase, is not to be printed. If this operand is omitted, the program loads the FCB, skips to channel 1, prints the last 80 characters of the phase, and again skips to channel 1.

If the FCB is loaded from SYSIPT, a verification message cannot be defined in the buffer image phase.

**Note:** Loading an FCB image phase for horizontal copy control does not turn on the horizontal copy function of IBM 4248 Printers.

**UCB**

The UCB control statement is used to load the UCB of a printer. For 4248 printers, use the SYSBUFFLD control statement BANDID.



**SYSxxx**

Identifies the printer whose UCB is to be loaded. The printer must be a PRT1 device; SYSxxx must be SYSLST or a programmer logical unit; SYSxxx may be SYSLOG if for any reason, SYSLOG has to be assigned to a line printer.

**phasename**

Specifies the name of the cataloged phase which contains the required buffer image information.

**FOLD**

Indicates that the UCB is to be loaded with the folding operation code to cause printing of uppercase characters for lowercase bit combinations.

**NOCHK**

Suppresses data checks resulting from an attempt to print unprintable characters (during subsequent use of the printer, not during SYSBUFFLD).

**NULMSG**

Indicates that the 80-character verification message which follows the buffer image in the specified phase is not to be printed. If this operand is omitted, the program loads the UCB, skips to channel 1, prints the last 80 characters of the phases, and again skips to channel 1.

---

## Buffer Load Phases

The following standard UCB and FCB image phases are provided in the system.

### Standard Buffer Image Phases

IBM Printer	UCB		FCB
	Phase Name	Train Type	Phase Name
1403U	\$\$BUCB4	AN	-
3211 (PRT1)	\$\$BUCB	A11	\$\$BFCB
3203-5 (PRT1)	\$\$BUCB00	AN	\$\$BFCB00
3289-4 (PRT1)	\$\$BUCB10	64-character belt	\$\$BFCB10
3262 (PRT1)	\$\$BUCB22	64-character belt	\$\$BFCB22
4245 (PRT1)	See Note	See Note	\$\$BFCB23
4248 (PRT1)	See Note	See Note	\$\$BFCB
4248 (Native) and 6262	See Note	See Note	\$\$BFCBWM

*Figure 23. Standard Buffer Load Phases*

**Note:** For these printers, the correct UCB for the mounted train is loaded automatically by microcode.

The standard FBC image phases are designed for 12 inch forms and a line density of 6 lines per inch (lpi), with:

- Channel 1 on line 5
- Channel 9 on line 56
- Channel 12 on line 66
- End of page on line 72.

### Additional UCB Images

The following additional UCB images (including copies of the standard images - highlighted in the table) are supplied in object format:

## UCB Images

Table 8. Additional UCB Images

IBM Printer	Module Name	Train Type
1403U	IJBTRAN IJBTRGN IJBTRONA IJBTRPAN IJBTRPHN IJBTRPN IJBTRQNC IJBTRQN JBTRRN IJBTRSN IJBTRTN IJBTRYN IJBTRALA	AN or HN GN ONA PCS-AN PCS-HN PN QNC QN RN SN TN YN ALA
3203-5	IJBTVAN IJBTVGN IJBTVOAA IJBTVOAB IJBTVODA IJBTVONA IJBTVPAN IJBTVPHN IJBTVPN IJBTVQNC IJBTVQN IJBTVRN IJBTVSN IJBVTN IJBTVYN IJBTVALA	AN or HN GN OAA OAB ODA ONA PCS-AN PCS-HN PN QNC QN RN SN TN YN ALA
3211	IJBTRA11 IJBTRG11 IJBTRH11 IJBTRP11 IJBTRT11	All G11 H11 P11 T11
3262	IJBNA48 IJBNA64 IJBNA96 IJBNAHI	48-character 64-character 96-character High-performance 63-character set
3289-4	IJBBE48 IJBBE64 IJBBE96 IJB116CF  IJB128KK	48-character 64-character 96-character 116-character (Canadian French) 128-character (Katakana)

These additional UCB images must be link-edited before you load them. Any valid phasename may be assigned to them.



---

## Automatic Buffer Loading During IPL

The IPL routine automatically loads the UCB and/or FCB of each operational printer with the standard image phases for the device, for example with \$\$BUCB00 and \$\$BFCB00 for the 3203-5 printer. If you normally have some other train or belt mounted on the printer(s), link-edit the appropriate object-type image, so that automatic UCB loading can use the correct information. For example, if you normally use a TN chain on a 1403U printer, link-edit IJBTRTN with the phase name \$\$BUCB4.

For support (at IPL) of the dualling feature and of trains/belts and forms not covered by the standard buffer image phases, generate your own images phases as described below and catalog them under the standard phase names.

### Creating Your Own UCB/FCB Image Phases

If you use non-standard trains or belts on your printer, or if you use special forms, you must:

1. Create the necessary UCB/FCB image phases, using the information given in Figure 24 on page 360;
2. Assemble and link-edit the new phases;
3. Ensure that the phases are stored on an accessible sublibrary.

If they are to be loaded automatically at IPL, link-edit them with the phase names of the standard images. Catalog them in the system sublibrary IJSYSRS.SYSLIB.

## UCB/FCB Phase Formats

IBM Printer	Buffer	Bytes	Contents
1403U	UCB	1-240 241-320	Train image Verification message
3211 (PRT1)	UCB	1-432 433-447 448-511 512 513-592	Train image Zeros Associative field Zero Verification message
	FCB (no indexing byte)	1-255 or 1-112, or 1-180, or 1-192 256-335	FCB image  Verification message
	FCB (with indexing byte) See Note	1-256, or 1-181 257-336	FCB image Verification message
3203-5 (PRT1)	UCB	1-240 241-304 305-512 513-592	Train image Associative field Zeros Verification message
	FCB	1-255, or 1-112, or 1-180, or 1-192 256-335	FCB image  Verification message
3262 (PRT1)	UCB	1-288 289-512 513-592	Belt image Zeros Verification message
	FCB	1-255, or 1-112, or 1-180, or 1-192 256-335	FCB image  Verification message

Note: If the indexing control byte is specified in the FCB image phase for a PRT1, it is used only if the printer is a 3211; otherwise, it is ignored.

Figure 24. Formats of UCB/FCB Image Phases (Part 1 of 2)

IBM Printer	Buffer	Bytes	Contents
3289-4 (PRT1)	UCB	1-256 257-512 513-592	Font offset table Zeros Verification message
	FCB	1-255, or 1-112, or 1-180, or 1-192 256-335	FCB image  Verification message
4245 (PRT1)	FCB  FCB FCB	256-335 1-255, or 1-112, or 1-180, or 1-192 256-335	Verification message FCB image  Verification message
4248 (PRT1 Mode)	UCB	1-432 433-447 448-511 512 513-592	Train image Zeros Associative field Zero Verification message
	FCB (no indexing byte)	1-255 or 1-112, or 1-180, or 1-192 256-335	FCB image  Verification message
	FCB (with indexing byte)	1-256, or 1-181 257-336	FCB image  Verification message
4248 (Native Mode*)	FCB	1-260 261-340	FCB image Verification message
6262**	FCB	1-260 261-340	FCB image Verification message

\* For the control characters to be used in the phase, see the IBM publication *4248 Printer Model 1 Description*, GA24-3927.

\*\* For the control characters to be used in the phase, see the IBM publication *6262 Printer Models 012 and 014 Product Description* GA24-4134.

Figure 24. Formats of UCB/FCB Image Phases (Part 2 of 2)

Legend:

**Train image**

The hexadecimal equivalent of all characters on the train (chain or belt).

**FCB image**

Control characters for the FCB, as defined in "FCB Characters" on page 362.

**Font offset table**

Table containing one entry for each possible hexadecimal combination from X'00' to X'FF'. Each entry contains the hexadecimal displacement of the

## UCB/FCB Phase Formats

appropriate printed character from the home position (X'00') of the belt (or one of the home positions if the character set is repeated on the belt).

The home position has a displacement of X'00'. Unused hexadecimal combinations have a displacement of X'80' and cause a data check on printing. Combinations X'00' (null) and X'40' (blank) have a displacement of X'7F' and suppress printing at the corresponding position on the line.

### Associative field

This is used for suppressing invalid characters. For further details, see the hardware description of the respective printers.

### Verification message

An 80-byte message which is printed after the load (unless NULMSG is specified). This message must be included in the image phase (even if all 80 bytes contain X'40').

---

## Loading the FCB Using SYSIPT

When the FCB is loaded using SYSIPT, there is no verification message. You supply the FCB image phase in card format immediately behind the FCB SYSxxx control statement. Each card column corresponds to a line on the forms to be used, that is card 1, column 1 refers to line 1; card 2, column 1 refers to line 81, and so on. The codes to be punched (or written to a diskette) are described in "FCB Characters."

Note that the 3211 indexing control byte cannot be specified if the FCB is loaded using SYSIPT. FCBs for IBM 4248 Printers cannot be loaded using SYSIPT.

## FCB Characters

The FCB characters for the phase and SYSIPT formats are shown in the table below.

Channel	Phase format (Hex)	SYSIPT punch code
None	00	blank
1	01	1
2	02	2
3	03	3
4	04	4
5	05	5
6	06	6
7	07	7
8	08	8
9	09	9
10	0A	A
11	0B	B
12	0C	C
End of FCB	10	X
8 lines per inch	10	*

### End of FCB

In the phase format for a PRT1 printer, it is possible to combine a channel character and the end-of-FCB character in the last buffer position used, if these two conditions coincide. For example, if channel 12 is on the last line of the form, the X'10' for end-of-FCB and the X'0C' for channel 12 can be combined by coding X'1C'. This is not possible for a non-PRT1 printer, nor is it possible when the FCB is loaded using SYSIPT.

**8 lines per inch**

This can be specified only for a PRT1 printer; all other printers have a hardware switch for selection of line density. If used, the \* (SYSIPT format) or X'1x' (phase format, where x may be a channel specification character) must be specified in the first column of the first (or only) card or in the first buffer position, respectively.

**Examples of FCB Image Phases**

1. Example of the source code for a PRT1 printer FCB image phase (LPI=6, paper size=12 inches, used FCB positions: 12x6=72):

```
// JOB FCB6PRT1
// OPTION CATAL
  PHASE FCB6PRT1,*
// EXEC ASMA90,SIZE=ASMA90
  START 0
  DC   XL4'00'           FCB POSITIONS 1 TO 4
  DC   X'01'            CHANNEL 1 ON LINE 5
  DC   XL24'00'         FCB POSITIONS 6 TO 29
  DC   X'05'            CHANNEL 5 ON LINE 30
  DC   XL41'00'         FCB POSITIONS 31 - 71
  DC   X'1C'            END OF FORMS AND CHANNEL 12
*                               ON LINE 72
  DC   XL183'00'        FCB POS. 73 - 255 ZEROS
  DC   CL80'PHASE FCB6PRT1 LOADED'
*                               LPI=6, PAPER SIZE=12 INCHES,
*                               CHANNEL 1/5/12 ON LINE 5/30/72
  END
/*
// EXEC LNKEDT
/ &
```

2. Example of a PRT1 printer FCB image phase (LPI=8, paper size=12 inches, used FCB positions: 12x8=96):

```
// JOB FCB8PRT1
// OPTION CATAL
  PHASE FCB8PRT1,*
// EXEC ASMA90,SIZE=ASMA90
  START 0
  DC   X'10'            LPI=8
  DC   XL3'00'          FCB POSITIONS 2 TO 4
  DC   X'01'            CHANNEL 1 ON LINE 5
  DC   XL54'00'         FCB POS. 6 TO 59
  DC   X'09'            CHANNEL 9 ON LINE 60
  DC   XL29'00'         FCB POS. 61 TO 89
  DC   X'0C'            CHANNEL 12 ON LINE 90
  DC   XL5'00'          FCB POS. 91 - 95
  DC   X'10'            END OF FORMS ON LINE 96
  DC   CL159'00'        FCB POS. 97 - 255 ZEROS
  DC   CL80'PHASE FCB8PRT1 LOADED'
*                               LPI=8, PAPER SIZE=12 INCHES,
*                               CHANNEL 1/9/12 ON LINE 5/60/90
  END
/*
// EXEC LNKEDT
/*
```

## FCB Phase Examples

---

## Chapter 8. Maintain System History Program (MSHP)

The control statements of the Maintain System History Program (MSHP) are of two types:

- Function control statements (summarized in Table 9 on page 366), which are used to define to MSHP the required function.
- Detail control statements (summarized in Table 10 on page 367), which are used to provide further details about the requested function.

To use an MSHP function, build a job or job step comprising:

- Job control ASSGN statements or commands for the necessary logical units as given in the description of the function control statement;
- The job control statement or command:  
`[//] EXEC MSHP`
- The applicable MSHP function control statement;
- Depending on the requested function, one or more detail control statements.

The two types of MSHP statement are described under separate headings and in alphabetical order. Each description includes:

1. The syntax notation.
2. A general description of the purpose and function of the statement, plus any special considerations and restrictions.
3. For function control statements, the system and programmer logical unit assignments required for the specific MSHP function. These assignments include work files used by MSHP or by any other system program that is invoked by MSHP (for example, assembler, librarian, or linkage editor). MSHP requires the same logical unit and extent information as the called program.
4. For function control statements, any required or optional detail control statements.
5. A detailed explanation of the statements' operands, together with any restrictions and the applicable default values.

MSHP control statements may be entered from SYSIN or SYSLOG.

Tables 9 and 10 give an overview of the function control statements and detail control statements, respectively. In these figures, the shortest valid form of each operation code is shown in capital letters. The effect of the statements is summarized under "Purpose".

**Note:** For examples on how to use MSHP to build distribution tapes, PTFs or APAR fixes, see the manual *Preparing a Product for VSE*.

## MSHP

Table 9. Function Control Statements - Overview and Purpose

Function Control Statement	Purpose
APply	Install a PTF and record it in the system history file of the operational system.
ARChive	Enter information relating to products, components, PTFs and local or APAR fixes into the history file.
BACKup	Copy an auxiliary or system history file from disk to magnetic tape for backup purposes.
COpy	Copy a history file from disk to disk.
CORect	Install a local or APAR fix.
CReate	Pre-format a history file and reserve space for the PERsonalize function (see below).
DUMP	Produce a formatted printout of a system or auxiliary history file.
INCorporate	Install a component distributed in SYSIN format.
INSTall SYsres/ PRoduct	Install a system (SYsres) or product (PRoduct).
INSTall SErvice/ BACkout	Apply preventive and corrective service from the service file (SErvice) or a backout tape (BACkout).
LIST	Retrieve information from a service file and write this information to SYSLST.
Lookup	Display on SYSLOG selected information from the system history file.
MERge	Insert entries of one history file into another history file.
PAth	Change a phase stored in a sublibrary.
PERsonalize	Identify the system history file in relation to a specific user.
REMOve	Erase entries from the system history file.
RESidence	Specify the names of the sublibraries in which a product resides.
REStore	Restore a complete shipment package or a history file from magnetic tape to disk.
RETRace	Retrieve information from the system history file and print the information on SYSLST.
REVoKe	Restore an operational system to the status that existed before the installation of a PTF.
SElect	Select individual tailor jobs from the generation file (for retailoring).
TAILOr	Identify and initiate the generation (or re-generation) of a sublibrary member.
UNdo	Remove an initiated local or APAR fix to re-establish the previous library status.



Table 10. Detail Control Statements - Overview and Purpose

<b>Detail Control Statement</b>	<b>Purpose</b>
AFFects	Specify the sublibrary members that are affected by a PTF or local fix application.
ALter	Specify text modifications for sublibrary members.
COMPAtible	Indicate the products that are compatible with the shipped product.
COMPRises	Identify the component, phases, modules, and/or macros that comprise a product, and enter the information in the history file.
DATA	Delimit input to the LIBR and LNKEDT programs.
DEFine	Create label/extent definitions for the history file.
DELete	Specify the lines to be deleted from a source book when applying a local fix.
EXCLude	Exclude one or more products, components, or PTFs from a service application.
EXECute	Call one or more system programs (for example, assembler) required for tailoring.
GENERate	Specify a phase, module, or macro for regeneration.
INCLude	Include one or more products, components, or PTFs in a service application.
INFLuences	Identify those generated phases, modules, or macros that are affected by a PTF or local/APAR fix and that have to be regenerated if the fix is applied.
INsert	Specify the lines to be inserted in a source book when applying a local fix.
INVolves	Explicitly request link-editing when installing a product or applying service.
OR	Delimit a set of requirements (initiated with the REQUIRES statement) and test the requirements.
PTF	List the PTFs whose cover letters are to be printed.
REPlace	Delimit where replacement lines for local or APAR fixes must begin and end; initiate the replacement of the source text.
REQUIRES	Specify the requirements for successfully installing a shipment package or applying service.
RESolves	Associate a comment with a PTF, a product, an APAR or a local fix, or a generated member.
REStart	Indicates, for macro updates, that a new sequence number series starts after the specified statement.
SCan	Scan a phase for a specified string, or display 16 bytes of a phase.
SUPERsedes	Record the PTFs that are superseded by a given PTF.
VERify	Specify where a verification is to be made for a local or APAR fix correction.

---

## High Level Assembler for VSE

With VSE/ESA 2.1 the DOS/VSE Assembler has been replaced by the High Level Assembler for VSE. This affects the TAILOR function and several control statements using a Type=E member type as default. E-macros are not directly supported by the High Level Assembler for VSE. Explicitly specify the type operand instead of using the default Type=E member type.

---

## Called System Control Programs

For certain functions, MSHP calls other VSE system control programs, such as the Librarian, the Linkage Editor, or the Assembler. These programs and their functions are described in the corresponding documentation of the programs and are not included in the MSHP Reference manual. For an overview of the VSE system and its components, see *z/VSE Guide to System Functions*.

---

## Types of History Files

MSHP works with two history files, the system history file and the auxiliary history file.

### System History File

The system history file is a permanent file that reflects the information about the parts contained in your system and the service applied to them. All changes made to your system via MSHP commands (such as INSTALL SERVICE, CORRECT, or INSTALL PRODUCT) are recorded in the system history file.

### Auxiliary History File

The term auxiliary history file has two different meanings in MSHP:

#### Alternate or second history file

All commands that directly address the history file distinguish between system history file and auxiliary history file. Some of these commands (like BACKUP HISTORY, DUMP HISTORY) allow to select the affected history file, others (like COPY HISTORY) use both history files at the same time.

#### Work file

Some of the MSHP commands (like BACKUP PRODUCT, LIST SERVICETAPE) need an internal work file (IJSYS02) for their processing. This work file may be a sequential disk file or located in VSAM-managed space.

**Note:** The auxiliary history file (work file) placed in VSAM-managed space does not use secondary allocations. Therefore the primary allocation must be large enough to contain the space required.

## Usage of MSHP Auxiliary History File

Table 11. Usage of MSHP Auxiliary History File

MSHP Function	Alternate History	Work File
BACKup History	X	
BACKup PProduct		X
COPy	X	
CORrect		X
CReate	X	
DUMP	X	
INSTall		X
LIST		X
MERge	X	
PERsonalize		X
REStore PProduct		X
REStore SYsres		X
REStore History	X	
TAILor		X

The auxiliary history file is not used for the following MSHP functions:

- APply
- ARChive
- INCorporate
- Lookup
- PAtch
- REMove
- RESidence
- RETRace
- REVoke
- SElect
- UNdo

## Restrictions

- The system history file and the auxiliary history file used as alternate or second history file cannot be placed in VSAM-managed space.
- The auxiliary history file used as work file cannot be placed in VSAM-managed space if the contents are to be reused by a following MSHP function control statement.

---

## MSHP Return Codes

On termination of any MSHP job step a return code will be set depending on the success of the invoked functions, or triggered by the occurrence of an error message. User jobs may utilize this information to control further processing. In addition, the return code is a hint to possible problems and a warning to check the job output for any error messages.

MSHP issues the following return codes:

**RC=0**

All commands and functions were processed as planned. No errors. No warning. Messages which were issued for information only will always set RC=0.

**RC=4**

Warning, no serious error. At least one function has not been processed because one assumption was not fulfilled. The result, however, is as expected.

Example:

- You tried to REMOVE a PTF or APAR which is not archived in the history file.

**RC=8**

One or more errors occurred during processing. This return code will be issued for errors where MSHP leaves it up to you to decide about further processing.

Examples:

- A component belonging to a product was not found in the history file.
- Application of a PTF was rejected.
- During execution of the INSTALL SERVICE command (mass-application of PTFs), none of the PTFs in the PTF file was applicable. Message NO PTF HAS BEEN APPLIED appears.

**RC=16**

This return code appears when a serious error makes all further processing useless. MSHP processing will terminate, and the job will also terminate if no JCL ON-statement is in effect.

The return codes will always be issued as long as the MSHP program is neither canceled nor abnormally terminated.

In many messages the return code will depend on the type of input. If MSHP is called from SYSLOG, there is a possibility to correct an MSHP control statement in error, or to make a decision for further processing. This is not possible for MSHP input read from SYSIPT. In these cases MSHP has to terminate or skip a function. Therefore a higher return code will be given.

---

## Repairing the History File

The following job uses the merge function to repair the internal structure of the history file. It first copies the history file to an auxiliary history file, and then merges the auxiliary history file back to the system history file. Use the MSHP BACKUP HISTORY function to backup the history file before running this job.

```

// JOB REPHIST
// DLBL IJSYS02,'WORK.HIST.FILE'
// EXTENT SYS018,SYSWK1,1,0,900,75
// ASSGN SYS018,DISK,VOL=SYSWK1,SHR
* CHECK THE ABOVE DLBL IF IT REFLECTS THE WORK HISTORY FILE
* ON YOUR SYSTEM, IT IS SET UP FOR 3390'S.
* WE SUGGEST TO BACKUP THE HISTORY FILE BEFORE YOU RUN
* THIS JOB.
// PAUSE
// EXEC MSHP
CREATE HIST AUX
COPY HIST SYS AUX
CREATE HIST SYS
MERGE HIST AUX SYS
/*
/&

```

Figure 25. Repairing the History File

---

## Rules for Writing MSHP Control Statements

- With one EXEC MSHP statement or command, any number of function control statements can be specified.
- The function control statement you use determines which detail control statement(s) must or may follow.
- Detail control statements can follow only a function control statement or another detail control statement.  
If detail control statements must be submitted in a specific sequence, this is noted in the description for the function control statement.
- The operands of a control statement should be coded in the sequence as shown.
- MSHP control statements are of free form. The operation codes may begin in any position of the input line.
- An input line for MSHP control statements represents the first 72 characters of a card image input record, or 120 characters for console input.
- Operation codes and operand keywords may be abbreviated. In the statement descriptions, permissible minimum abbreviations are shown as uppercase character strings, followed by the remainder of the keyword in lowercase. For example, INSTall may be coded as INST, INSTALL, or anything within these limits.
- A value contained within brackets [...] may be included or omitted, depending on the requirements of the program. Two or more values contained within brackets and separated by an | sign represent alternatives, one (and only one) of which may be chosen. For example:

**[IRRevokable | REVokable]**

In the example, **IRRevokable** is the default, which MSHP assumes if you enter nothing.

- Options contained within braces {...} and separated by an | sign represent alternatives, one of which *must* be chosen. For example:  
{PRODUCT | SYSres}
- The operands of a statement are separated from one another by:
  - One or more blanks

- A comment (which is text within /\* and \*/)
- A comma (which may be surrounded by one or more blanks or comments)
- An all-comment input line is allowed. However, it must not begin in column 1.
- Words given in all lowercase letters represent information that must be supplied by the user.
- The equal sign (=), the plus sign (+), the colon (:), and the single quotes ( ' ') must be coded as shown; they may be surrounded by one or more blanks, except for the (+) sign, which must *not* be preceded or followed by a blank.
- An ellipsis (a series of three periods) indicates that a list of up to 100 items (such as PTF numbers) may be specified within parentheses. For example:  
(UD27484,UD13528,...)

However, a single item does not have to be enclosed in parentheses.

- The individual values in a list can be separated from each other by:
  - One or more blanks
  - A comment (text within /\* and \*/)
  - A comma (which may be surrounded by one or more blanks or comments)

Commas and blanks as separators may be intermixed in a given list.

- A control statement (function or detail) ends with the end of the input line, unless it is explicitly continued by means of a dash (-), followed by at least one blank. It may also end with a semicolon.

The continuation dash must also be preceded by at least one blank, except after a

- Comma
- Parenthesis
- Equal sign
- Comment
- Quoted string.

For function control statements, not more than six continuation lines may be specified.

A pair of values connected by a colon, as in

```
APPLY 5686-CF7-07-81C:UD12345
```

cannot be broken by a line end; nor can a keyword itself, a number, or a string (with or without quotes) be continued on a subsequent line.

- From the console, MSHP control statements may be entered in uppercase or in lowercase.
- An MSHP statement entered from the console may be canceled by entering two question marks (??).

---

## Coding Conventions for Frequently Used MSHP Operands

### component

The term 'component' stands for the component identification number (or program number) of a component. Example:

```
5686-CF7-06
```

A component may occur in more than one product, in which case it is further qualified by a 'level' indication.

**level**

This is a string of three alphanumeric characters which identifies the component uniquely if, for example, the component is shared by several different licensed programs.

To indicate that a component belongs to a certain program, the level number of that program is hyphenated with the component name.

For example:

5686-CF7-06-81C

identifies component 5686-CF7-06 at level 81C.

**product**

The term 'product' stands for the 6-character identification number of a licensed program, for example:

CF781C

The first three characters (CF7 in this example) are the product code. This is derived from the program number. The remaining characters (81C in this example) are the level number of the program, formerly known as feature number or release number.

This level number is also the level number of any component(s) belonging to the program.

MSHP supports multiple levels of a program. A component can be installed several times with an identical product code but different levels.

**apar-number**

A string of seven characters, consisting of two alphabetic characters, followed by five digits. Example:

DY12345

**ptf-number**

A string of seven characters consisting of two alphabetic characters, followed by five digits. Example:

UD12345

**Note:** The lib and sublib operands follow the syntax of the librarian program (for details, refer to Chapter 5, "Librarian," on page 285).

**lib**

The name of a library.

**sublib**

The name of a sublibrary.

**Note:** Do not use \$\$MSHPxx as a sublibrary name.

**member-name**

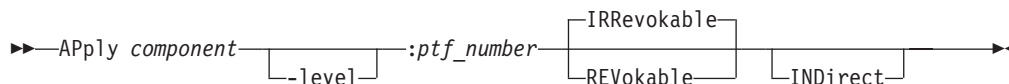
The name of a sublibrary member (phase, module, or macro). It consists of one to eight alphanumeric characters, the first of which must be alphabetic.

**member-type**

This member-type can be one character only or PROC or HTML.

## Function Control Statements

### APPLY



The APPLY statement is used to install a single PTF to your system and to record the installation in the system history file.

#### Logical Unit Assignments

Required:

##### SYS001

Work file used by MSHP.

##### SYSLNK

Linkage editor input file; needed to catalog phases supplied by IBM in object format.

##### SYSLST

System printer.

Optional:

##### SYSPCH

Needed if a backout PTF is to be generated (via the REVOKABLE operand).

##### SYSxxx

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

#### Related Detail Control Statements

Required:

AFFECTS

DATA

RESOLVES 'comment' APARS=(apar-number,...)

Optional:

DEFINE

INFLUENCES

INVOLVES

OR

REQUIRES

SUPERSEDES

#### Description of Operands

##### component[-level]

Specifies the component to which the PTF is to be installed.

Level specifies the level number (formerly release number) of the applicable component.

If the level number is not specified, application of the PTF depends on how many levels of the component are installed. If there is only one level installed, the PTF is applied to this one; otherwise, MSHP informs you which levels are installed and asks you to which one you want to apply the PTF.



**ptf-number**

Specifies the number of the PTF to be installed.

**IRRevokable**

Specifies that, when installing the PTF, MSHP will not produce any backout PTF jobs. The PTF cannot be revoked, that is, the status before the installation of the PTF cannot be recreated at a later point in time.

**REvokable**

Specifies that, when installing the PTF, a backout PTF job (with a REVOKE statement) is to be generated on SYSPCH. This allows you to recreate the status of your system as it existed before the installation of the PTF.

Restrictions:

- If SYSPCH is assigned to tape, the backout job can later be started by assigning SYSIN to that tape, should this become necessary. However, the tape cannot be processed with the INSTALL BACKOUT statement.
- Do not specify REVOKABLE for a PTF that is a pre- or co-requisite for other PTFs or has comparable local/APAR fix dependencies.
- Do not specify REVOKABLE if the PTF contains new or additional modules or macros that are not part of the current component release.

**INDirect**

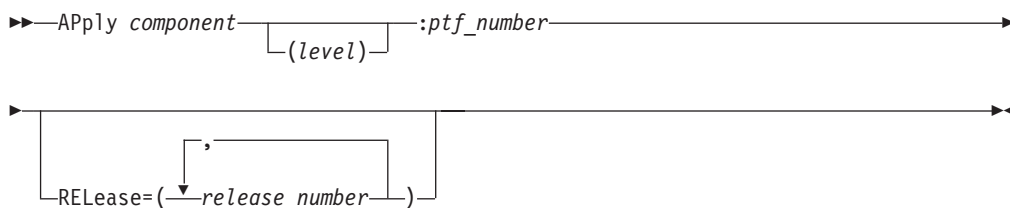
Specifies a PTF for indirect application via the Service Dialog of z/VSE. The operand indicates to the INSTALL SERVICE SD function that the sublibrary members affected by the service application are first to be applied to a reserved sublibrary \$\$MSHPIL before they are finally moved to the system sublibrary IJSYSRn.SYSLIB. This is to protect the IPLed SYSLIB in case the PTF application fails.

**Example:** APPLY 5686-CF7-07-81C : UD19345

Detail Control Statements:

```
RESOLVES 'comment' APARS=(DY50001,DY50010)
AFFECTS MODULES=(IKRUPGR,IKRINSTL)
DATA
```

For compatibility reasons, the APPLY statement is still accepted in the following format:



**(level)**

Indicates the old three-character alphanumeric feature identifier of the component.

If this operand is specified, any following release information is ignored.

**RELease=(release-number,...)**

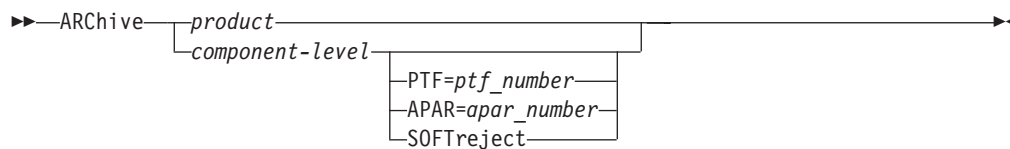
Specifies the release(s) of the component to which the PTF is to be installed.

## APPLY

This operand applies to old-format statements only and is ignored if level was specified (see above). If level was not specified, MSHP converts the release number into a level number.

If neither level nor release is specified, application of the PTF depends on how many levels (releases) of the component are installed. If there is only one level installed, the PTF is applied to this one; otherwise, MSHP informs you which levels are installed and asks you to which one you want to apply the PTF.

## ARCHIVE



The ARCHIVE statement is used to make entries in the system history file.

### Logical Unit Assignments

Required:

#### SYSLST

System printer.

Optional:

#### SYSxxx

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

The detail control statements related to the operands of the ARCHIVE statement are listed in the following table. Required detail statements are marked with an 'R', optional detail statements with an 'O':

Table 12. Detail Control Statements Related to ARCHIVE Operands

ARCHIVE...	Prod.	Comp.	PTF	APAR
AFFECTS			R	R
ALTER				O
COMPRISES	R			
DEFINE	O	O	O	O
DELETE				O
INSERT				O
INVOLVES	O	O	O	
OR	O	O	O	
REPLACE				O
REQUIRES	O	O	O	
RESOLVES	R		R	O
SUPERSEDES			O	

### Description of Operands

#### product

Specifies that an entry for the named product is to be made in the system history file.

Note that the components comprised in an archived product have to be archived, too, before other functions (for example, RESIDENCE) can be executed for this product entry.

## ARCHIVE

### **component-level**

Specifies that an entry for the named component is to be made in the system history file.

If a PTF or APAR is specified, 'component' identifies the component to which the particular PTF, local fix, or APAR fix to be archived applies.

### **PTF=ptf-number**

Identifies the PTF for which an entry is to be made in the history file.

### **APAR=apar-number**

Identifies the local or APAR fix for which an entry is to be made in the history file.

### **SOFTreject**

Specifies that a PTF which may have to be installed to the named component is to be installed even if, as a result, a local or APAR fix would be partially overwritten. (The same applies to a PTF that may have to be revoked.) For a component that is archived without SOFTREJECT specified, MSHP automatically rejects the installation (revocation) of a PTF that partially overwrites a local or APAR fix.

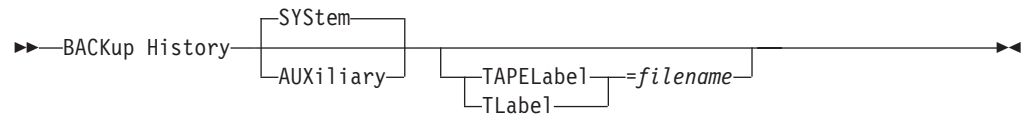
Use the option only if the result of a partial overwrite does not cause an immediate compatibility problem (as, for example, the replacement of asynchronously executed phases).

**Example:** ARCHIVE 5686-CF7-07-81C PTF=UD23453

Detail Control Statements:

```
AFFECTS MODULE=MODAAA  
RESOLVES 'comment' APAR=DY32555
```

## BACKUP HISTORY



The BACKUP HISTORY statement requests MSHP to copy a history file located on disk onto magnetic tape.

### Logical Unit Assignments

Required:

#### SYs006

The tape onto which MSHP writes the backup copy of the history file.

#### SYSLST

System printer.

Required for BACKUP HISTORY AUXILIARY:

#### SYsyyy

The device on which the auxiliary history file resides. This can be either SYs002 or the device specified in the UNIT operand of the DEFINE HISTORY AUXILIARY statement.

Optional:

#### SYsxxx

Required for BACKUP HISTORY SYSTEM if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required: none

Optional: DEFINE

### Description of Operands

#### SYStem

Specifies that the system history file is to be copied to tape.

#### AUXiliary

Specifies that the auxiliary history file is to be copied to tape.

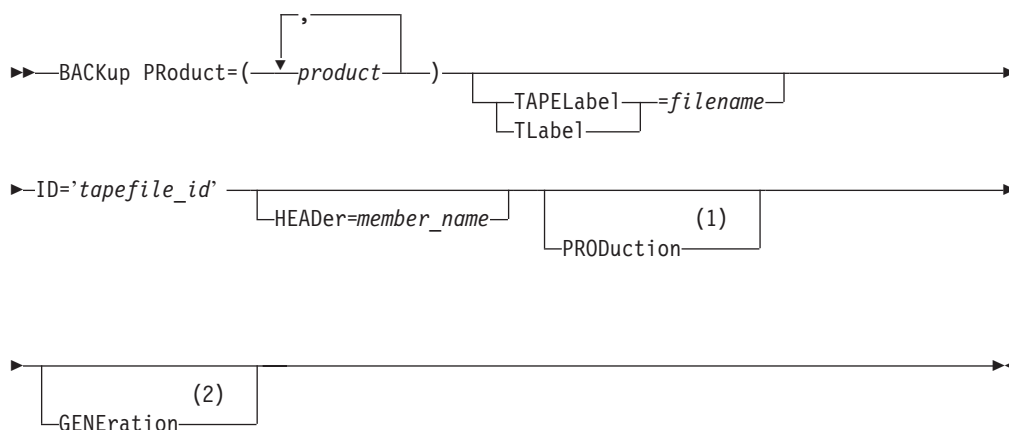
#### TAPeLabel | TLabel=filename

Specifies that the history file is to be copied to a tape with standard labels. **filename** is the seven-character filename that is specified in the // TLBL statement for the backup file.

**Example:** BACKUP HISTORY SYSTEM

## BACKUP PRODUCT

### BACKUP PRODUCT



#### Notes:

- 1 If neither PRODUCTION nor GENERATION is specified, both the production and generation sublibraries are copied.
- 2 If neither PRODUCTION nor GENERATION is specified, both the production and generation sublibraries are copied.

The BACKUP PRODUCT statement is used to produce, on magnetic tape, a backup copy of the named program(s), referred to in this section as product(s). This backup copy consists of the production and generation sublibraries of the product(s), together with the pertinent system history file containing product-related entries. You can later re-install the product(s) with the INSTALL PRODUCT statement.

**Note:** MSHP always copies a complete sublibrary, which also includes the products that are not specified in the BACKUP statement. Therefore, the backed-up history file reflects all products of the backed-up sublibrary.

#### Logical Unit Assignments

Required:

##### SYS006

The tape onto which MSHP writes the backup copy of the named product(s).

##### SYSLST

System printer.

##### SYSyyy

The device on which the auxiliary history file resides. This can be either SYS002 or the device specified in the UNIT operand of the DEFINE HISTORY AUXILIARY statement.

#### Related Detail Control Statements

Required:

None

Optional:

DEFINE

## Description of Operands

### **PRODUCT=(product,...)**

Specifies the product(s) for which a backup copy is to be produced.

All requested products must reside in the same set of production and generation sublibraries, since MSHP copies only entire sublibraries.

### **TAPELabel | TLabel=filename**

Specifies that the products are to be copied to a tape with standard labels.

**filename** is the seven-character filename that is specified in the // TLBL statement for the backup file.

### **ID='tapefile-id'**

Specifies the identifier of the backup file. MSHP uses this identifier to locate the backup file during RESTORE. The tapefile-id can be one to 16 alphanumeric characters, enclosed in quotes; it must not contain any quotes.

### **HEADER=member-name**

Specifies an additional sublibrary member that is to be written as a header file onto the backup tape (as the very first file created with this BACKUP statement). This header file can be used to write some informational text, or job control statements, or copyright information in front of the backup file.

MSHP searches for the denoted member under type 'Z' in the production sublibrary of the product to be backed up.

'member-name' denotes the name of the sublibrary member containing the header file information.

### **PRODUCTION**

Specifies that only the production sublibrary of the named product(s) is to be copied.

### **GENERATION**

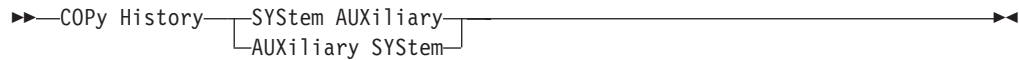
Specifies that only the generation sublibrary of the named product(s) is to be copied.

If neither PRODUCTION nor GENERATION has been specified, both the production and the generation sublibraries are copied.

**Example:** BACKUP PRODUCT=(099360) ID='DITTO.....1.3.0' PRODUCTION

See "Packaging Samples" in *Preparing a Product for VSE* how to backup a new product.

# COPY HISTORY



The COPY HISTORY statement requests MSHP to copy a history file from disk to disk.

### Logical Unit Assignments

Required:

#### SYSLST

System printer.

#### SYSyyy

The device on which the auxiliary history file resides. This can be either SYS002 or the device specified in the UNIT operand of the DEFINE HISTORY AUXILIARY statement.

Optional:

#### SYSxxx

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

none

Optional:

DEFINE HISTORY

### Description of Operands

#### SYStem AUXiliary

Creates a copy of the system history file for use as an auxiliary history file.

#### AUXiliary SYStem

Copies an auxiliary history file to the system history file.

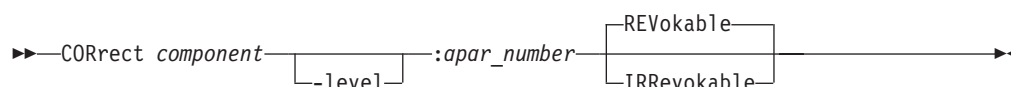
**Note:** If the new (copied) history file extent is to reside on a newly defined VM minidisk, this minidisk must have been initialized by:

- CMS command FORMAT, followed by
- VM disk initialization program IBCDASDI, or
- Device Support Facilities INIT command with parameter 'Mimic(Mini(n))'.

**Example:** COPY HISTORY SYS AUX



## CORRECT



The CORRECT statement specifies that a local or APAR fix is to be installed to a component. It is, however, not possible to install a local or APAR fix for members of type PROC and HTML.

**Notes:**

1. To avoid an unintended removal of a fix due to linkage editor or assembly runs after the application of the fix, a correction should always be made in all applicable sublibrary members (phases, modules, macros).
2. Since the High Level Assembler for VSE cannot create E-decks, they can only be changed by the CORRECT function if the old DOS/VSE Assembler is installed.

**Logical Unit Assignments**

Required:

**SYSLST**

System printer.

Optional:

**SYSLNK**

Linkage editor input file; needed if the correction requires link-editing.

**SYSPCH**

Needed when correcting a macro and REVOKABLE is specified.

**SYS001,****SYS004**

Needed as work files if the correction involves:

Modules,  
Expandable phases, or  
Macros.

**SYS002,****SYS003**

Needed as work files if corrections to macros are involved.

**SYSxxx**

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

**Related Detail Control Statements**

Required:

AFFECTS

Optional:

ALTER  
DEFINE HISTORY SYSTEM  
DELETE  
INFLUENCES  
INSERT  
INVOLVES  
REPLACE

## CORRECT

REQUIRES  
OR  
RESOLVES 'comment'  
RESTART  
SCAN  
VERIFY

The detail control statements must be entered in the following sequence:

1. DEFINE HISTORY SYSTEM
2. REQUIRES, OR
3. RESOLVES
4. AFFECTS
5. ALTER, DELETE, INSERT, REPLACE, RESTART, SCAN, and VERIFY, if used, must be coded after the AFFECTS statement.
6. INFLUENCES, INVOLVES.

### Description of Operands

#### **component[-level]**

Specifies the component that is to be corrected by the local or APAR fix.

If level is not specified, the application of the fix depends on how many levels of the component are installed. If only one level is installed, MSHP applies the fix to this one; otherwise, MSHP informs you which levels are installed and asks you for the requested one.

#### **apar-number**

Specifies the number of the local or APAR fix that contains the correction(s).

#### **REVokable**

Specifies that corrections made to phases or modules can be removed by using the UNDO function.

For corrections to macros, the REVOKABLE option causes a job to be created on SYSPCH with the initiating control statement:

```
UNDO component:apar-number
```

The correction data consists of catalog requests for the unaltered version of the macro(s), enclosed in DATA statement delimiters.

#### **IRRevokable**

Specifies that corrections cannot be revoked.

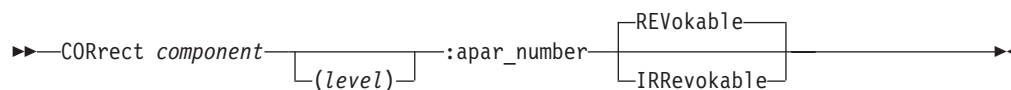
**Example:** CORRECT 5686-CF7-07-81C : DY21001

Detail Control Statements:

```
AFFECTS PHASE=MAINPHA  
SCAN 0730 ARG=4130A346  
ALTER 0730 4130A346 : 47F0C71C
```

See the chapter 'APAR Fix (ZAP)' in the *Preparing a Product for VSE* manual for more examples.

For compatibility reasons, the CORRECT statement is still accepted in the following format:

**(level)**

Indicates the old three-character alphameric feature identifier of the component.

If level is not specified, the application of the fix depends on how many levels of the component are installed. If only one level is installed, MSHP applies the fix to this level; otherwise, MSHP informs you which levels are installed, and asks you to specify the level to which the fix is to be applied.

## CREATE HISTORY



The CREATE HISTORY statement requests MSHP to initialize a history file. (For information on creating extent definitions, refer to the DEFINE HISTORY detail control statement.)

### Logical Unit Assignments

Required:

#### **SYSLST**

System printer.

Required for CREATE HISTORY AUXILIARY:

#### **SYSyyy**

The device on which the auxiliary history file resides. This can be either SYS002 or the device specified in the UNIT operand of the DEFINE HISTORY AUXILIARY statement.

Optional:

#### **SYSxxx**

Required for CREATE HISTORY SYSTEM if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

none

Optional:

DEFINE

### Description of Operands

#### **SYSem**

Specifies that a system history file is to be initialized.

#### **AUXiliary**

Specifies that an auxiliary history file is to be initialized.

**Note:** If the new (copied) history file extent is to reside on a newly defined VM minidisk, this minidisk must have been initialized by:

- CMS command FORMAT, followed by
- VM disk initialization program IBCDASDI, or
- Device Support Facilities INIT command with parameter 'Mimic(Mini(n))'.

**Example:** CREATE HISTORY SYSTEM

## DUMP HISTORY



The DUMP HISTORY statement requests MSHP to produce a formatted hexadecimal printout of a history file on SYSLST.

This statement is provided primarily as an aid for program service. IBM service personnel may ask you to use it to produce a dump for diagnosis purposes.

### Logical Unit Assignments

Required:

#### SYSLST

System printer.

Required for DUMP HISTORY AUXILIARY:

#### SYSyyy

The device on which the auxiliary history file resides. This can be either SYS002 or the device specified in the UNIT operand of the DEFINE HISTORY AUXILIARY statement.

Optional:

#### SYSxxx

Required for DUMP HISTORY SYSTEM if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

none

Optional:

DEFINE

### Description of Operands

#### SYStem

Specifies that the system history file is to be dumped.

#### AUXiliary

Specifies that the auxiliary history file is to be dumped.

**Example:** DUMP HISTORY AUX

## INCORPORATE



The INCORPORATE statement is used to install a component distributed in SYSIN format.

### Logical Unit Assignments

Required:

#### SYSLNK

Linkage editor input file.

#### SYS001

Linkage editor work file.

#### SYSLST

System printer.

Optional:

#### SYSxxx

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

DATA

Optional:

DEFINE

INVOLVES

OR

REQUIRES

### Description of Operands

#### component[-level]

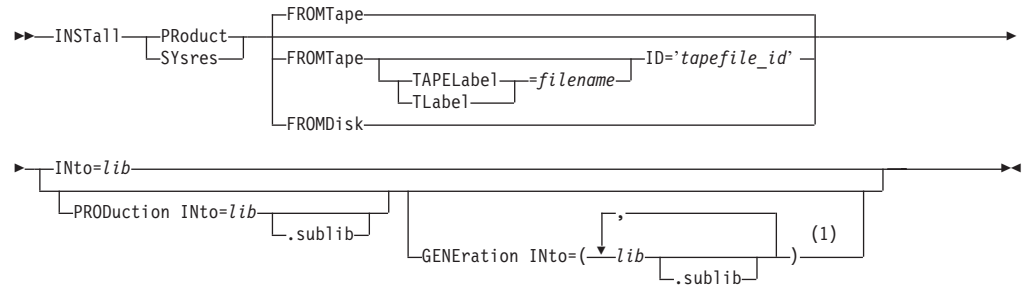
Identifies the component to be incorporated. If you specify a level, any following release information is ignored. If you specify component without level, you must indicate the release number of the component in the RELEASE operand.

#### RELease=release-number

Specifies the release of the component to be incorporated (only applicable if 'level' is not specified in the component operand). MSHP converts the release number into a level number.

**Example:** INCORPORATE 5686-CF7-06-81C

## INSTALL PRODUCT/SYSRES



**Notes:**

1 You can specify up to nine sublibraries

The INSTALL statement requests MSHP to install either a licensed program (referred to as product in this section), such as VSE/VSAM, or a SYSRES package, such as z/VSE.

The shipment history file that accompanies the software to be installed reflects the contents of the shipment package; it may contain information on any requirements that have to be met prior to installation (for example, pre-requisite components or PTFs). When executing the INSTALL function, MSHP informs you of any missing requirements.

MSHP restores the shipment history file from the distribution medium into an auxiliary history file. You may (1) either use the standard SYS002 work file for the auxiliary history file, or (2) specify user labels for IJSYS02, or (3) define it with a DEFINE HISTORY AUXILIARY statement.

With the restored auxiliary history file, checking for pre-, co-, and negative-requisites is performed. If all checks and verifications prove satisfactory, the distribution libraries are restored into the specified target libraries, and the restored distribution history file is merged with the current system history file.

MSHP also determines (by analyzing the shipment history file and your system's history file) which of the products already installed in your system are compatible with the shipped product and which are superseded:

- Products that are based on the same base products are usually *compatible* with each other. This relationship can also be explicitly defined via the COMPATIBLE detail control statement.
- An installed product is *superseded* when you install a follow-on release of that product. In that case MSHP (1) informs you that the new shipment package supersedes the current level of the product and (2) asks you whether you want to keep the old version of the product or delete it.

### Logical Unit Assignments

Required:

**SYS006**

Distribution file.

## INSTALL PRODUCT/SYSRES

### **SYSyyy**

The device on which the auxiliary history file resides. This can be either SYS002 or the device specified in the UNIT operand of the DEFINE HISTORY AUXILIARY statement.

Optional:

### **SYSxxx**

Required if the device on which the system history resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

## **Related Detail Control Statements**

Required:

none

Optional:

DEFINE HISTORY  
COMPATIBLE

## **Description of Operands**

### **PRoduct**

Specifies that a product (non-SYSRES package) is to be installed.

### **SYSres**

Specifies that a SYSRES package is to be installed.

### **FROMTape [TAPELabel | TLabel=filename] ID='tapefile-id'**

Indicates that the distribution tape is to be searched for the specified tapefile-id. This must be identical with the tapefile-id specified in the BACKUP statement. If the tape is not correctly positioned, it is scanned for the specified ID, and positioned correctly before installation. Note that the scan is only in forward direction and will stop at either the correct tapefile-id or at end-of-file (TM, EOB, TM), whichever occurs first.

The tapefile-id can be one to 16 alphameric characters.

If the operand is omitted, it is assumed that the tape is correctly positioned to the product to be installed.

### **TAPELabel | TLabel=filename**

Specifies that the tape contains standard labels. **filename** is the seven-character filename that is specified in the // TLBL statement for the tape.

### **FROMDisk**

FROMDisk must be specified to support the INSTALL function for a system without magnetic tape.

As a preparatory step, the sublibraries and the shipment history file on the distribution medium must have been restored to disk with the RESTORE PRODUCT/SYSRES statement.

The restored history file must then be made known to the INSTALL FROMDISK function with a DEFINE HISTORY AUXILIARY detail control statement. The disk with the auxiliary history file must be assigned to SYS002, or it must be specified in the UNIT=SYSnnn operand of the DEFINE statement.

With the restored history file, all necessary checks and verifications are performed; if they are satisfactory, the distribution libraries are merged into the specified target libraries, and the restored distribution history file is merged with the current system history file.



**INto=...**

Specifies the names of the libraries and sublibraries into which the members from the distribution file are to be copied. Via the parameters PRODUCTION and GENERATION you may indicate that you want to install either the production part or the generation part of the shipment package only, or install the two parts into separate libraries.

**Rules for Target Libraries/Sublibraries:**

- Product's first installation: The product is not yet defined or has been removed from the history file. If you specify neither a library nor a sublibrary, MSHP takes one of the following as target library:
  1. If the shipped product supersedes another one, the library and sublibrary of the superseded product.
  2. Otherwise, the library and sublibrary of a compatible product, if there exists any.
  3. If none of the above, MSHP notifies you and terminates installation.

If you specify a target library, but no sublibrary, MSHP takes one of the following:

1. The sublibrary of any superseded product with the same library.
  2. Or the sublibrary of any compatible product with the same library.
  3. If none of the above, the sublibrary name of the shipped production and/or generation sublibrary.
- Product reinstallation: The shipped product is already installed as, for example, in the case of a refresh installation. MSHP takes the library and sublibrary of the installed product, regardless of whether you have specified a different library and/or sublibrary. In this case, a decision message will be issued, asking you for confirmation.
  - Installation of SYSRES: If, for INSTALL SYSRES, you do not specify a sublibrary (sublib) name, MSHP uses the name of the shipment sublibrary, which is SYSLIB. If the target libraries (lib) do not exist, MSHP creates them. However, you must provide the necessary label information (DLBL/EXTENT) for the libraries.

**INto=lib**

Specifies, for INSTALL PRODUCT only, installation of both the production and the generation part of the shipment package into the library denoted by 'lib'.

**Note:** This operand cannot be used for INSTALL SYSRES, because the generation part and the production part must be installed into different target libraries.

**PRODUCTION INto=lib[.sublib]**

Specifies installation of the executable (production) part of the shipment package, which consists of all phases, procedures (and some modules/macros) needed for daily operation of your system or product. The production part must be installed before the generation part.

MSHP merges the members of the production shipment sublibrary into the target sublibrary indicated by lib[.sublib] or, if sublib has been omitted, into the sublibraries established by MSHP (see *Rules for Target Libraries/Sublibraries* on page 391).

For INSTALL SYSRES, however, the name of the production library must always be specified as IJSYSRn, n being a digit from 1 to 9. The generation part must be installed into a different library than the production part.

## INSTALL PRODUCT/SYSRES

### GENERation INto=(lib[.sublib],...)

Specifies installation of the generation part of the shipment package, which contains those modules and, possibly, macros that are needed for the regeneration of the product.

MSHP merges the members of the shipped generation sublibrary into the target sublibrary (or sublibraries) indicated by (lib[.sublib],...), or if sublib has been omitted, into the sublibraries established by MSHP (see *Rules for Target Libraries/Sublibraries* on page 391).

### Examples:

```
INSTALL PRODUCT FROMTAPE ID='LM4E11' -  
        INTO=USER01
```

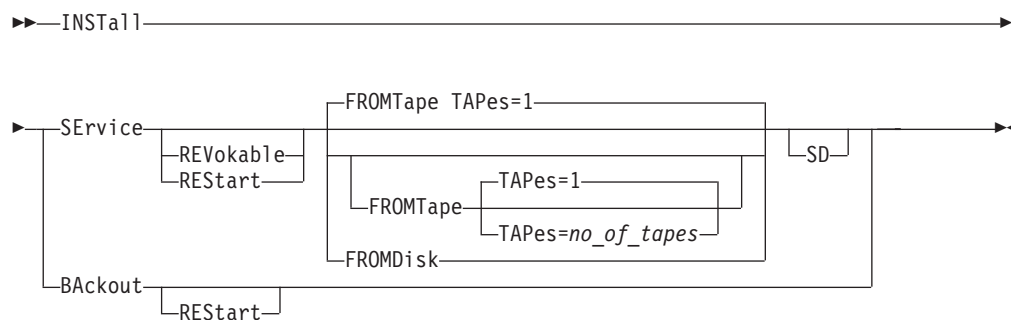
```
INSTALL PRODUCT FROMTAPE ID='LM4E11' -  
        PRODUCTION INTO=USER01.PRODALL
```

```
INSTALL PRODUCT FROMDISK -  
        PRODUCTION INTO=USER01.PRODALL
```

```
INSTALL SYSRES FROMTAPE ID='CF781C' -  
        PRODUCTION INTO=IJSYSR1
```

For more examples see the manual *Preparing a Product for VSE* under "Installing a Product or Feature".

## INSTALL SERVICE/BACKOUT



The `INSTALL SERVICE` statement requests MSHP to install PTFs from one or more service tapes or from the service file (which is a SAM file in VSAM-managed space).

The `INSTALL BACKOUT` statement requests MSHP to install one or more backout PTFs, which means recataloging the sublibrary member(s) replaced by installing the corresponding PTF(s). The statement works in the same way as the `INSTALL SERVICE` statement, except that it reads the PTF information from the backout tape, which is created when you specify the `REVOKABLE` operand in the `INSTALL SERVICE` statement. `INSTALL BACKOUT` does not support backout jobs that were created with the `REVOKABLE` option of an `APPLY` single PTF statement.

Via the `INCLUDE` and `EXCLUDE` detail control statements you can specify that only certain products, components, or PTFs are to be included or excluded during the service application. If you omit the `EXCLUDE` and `INCLUDE` statements, all service tape PTFs which are applicable to your system will be selected for service installation.

MSHP prints a list of all PTFs that are to be installed and asks you for confirmation before it replaces the affected members in your sublibraries and updates the history file.

### Logical Unit Assignments

Required:

#### **SYS006**

Service tape (not required for `FROMDISK`) / Backout tape.

#### **SYS001,**

#### **SYS002,**

#### **SYS003**

Work files used by MSHP.

#### **SYSLST**

System printer.

Optional:

#### **SYSxxx**

Required if the device on which the system history file resides is specified in the `UNIT` operand of the `DEFINE HISTORY SYSTEM` statement.

## INSTALL SERVICE/BACKOUT

### SYS004

Needed if backout PTF jobs are to be created (by specifying the REVOKABLE operand).

### Related Detail Control Statements

Required:

none

Optional:

INCLUDE

EXCLUDE (for INSTALL SERVICE only)

### Description of Operands

#### SERvice

Specifies that MSHP is to install PTFs from the service tape(s) or service file, as detailed in any INCLUDE or EXCLUDE detail control statements.

#### REVOkable

Specifies that backout jobs are to be created for all PTFs that are to be installed. The backout jobs are MSHP jobs with the REVOKE function control statement included. The backout jobs are written in blocked format onto a tape which must be assigned as SYS004. This tape can later be used as input for the INSTALL BACKOUT statement to re-install the PTFs, if necessary.

#### BACKout

Specifies that MSHP is to install one or more backout PTFs from the backout tape which is produced by the INSTALL SERVICE function with the REVOKABLE option. The PTFs to be installed can be selected via the INCLUDE statement.

PTFs with common sublibrary members are grouped together into one single PTF which contains all the members of the PTFs. In that case, a separation via the INCLUDE statement is not possible at INSTALL BACKOUT time.

#### REStart

Requests MSHP to restart a previous INSTALL SERVICE/BACKOUT or APPLY/REVOKE job whose final link step failed. MSHP scans the history file entries for those PTFs that were correctly cataloged, but not yet linked, and invokes the linkage editor to complete the final link step.

#### FROMTape

Specifies that MSHP is to install PTFs from the service tape(s). This operand is identical with the SERVICE operand; it is included for compatibility reasons (with FROMDisk) only.

#### TAPes=no.-of-tapes

Required only if two or more tapes are to be processed. Indicates to MSHP the number of tape volumes that have to be scanned for the particular service installation. If you know that prerequisite PTFs exist on other service tapes and that these PTFs are not yet installed, have MSHP scan these additional tape volumes for the prerequisite PTFs and have them retrieved for installation.

Mount the first tape on the tape drive assigned to SYS006 before you enter the EXEC MSHP command or statement. MSHP scans this tape and then issues message M363D, prompting you to mount the next tape on the same tape drive. When the last tape has been scanned, MSHP processes it and then issues message M363D again. You must now mount the first tape again, this time for processing. After processing each tape, MSHP issues message M363D, prompting you for the next tape.

For example, if you specify TAPES=3, the sequence of events is:

```

Mount tape 1;
MSHP scans tape 1;
Mount tape 2;
MSHP scans tape 2;
Mount tape 3;
MSHP scans and processes tape 3;
Mount tape 1 again;
MSHP processes tape 1;
Mount tape 2 again;
MSHP processes tape 2.
    
```

The maximum number that can be specified is 9. If the operand is omitted, *one* tape volume is assumed.

### SD

This operand indicates that service is to be applied via the z/VSE Service Dialog. For those PTFs that are flagged with the INDIRECT option (in the APPLY statement), the members affected by the service application are first applied to a reserved sublibrary \$\$MSHPIL before they are finally moved into the system sublibrary IJSYSRn.SYSLIB. This is to protect the IPLed SYSLIB in case the PTF application fails.

### FROMDisk

Specifies that MSHP is to apply PTFs from the service file on disk. The file must be defined with the following // DLBL statement:

```
// DLBL IJSYSPF, 'PTF.FILE', ,VSAM,CAT=VSESPUC
```

### Examples:

```

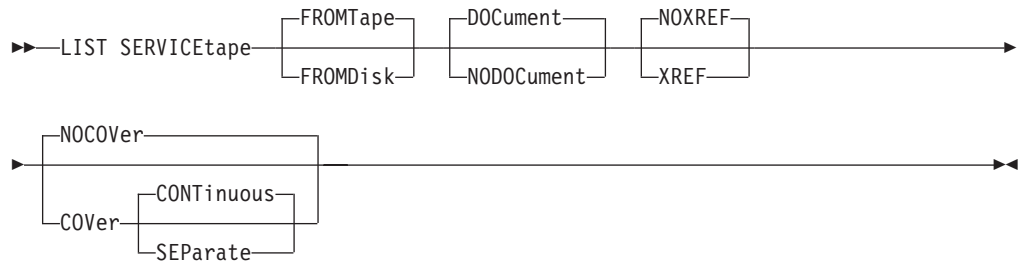
INSTALL SERVICE (default FROMTAPE is assumed)
INSTALL SERVICE FROMDISK
INSTALL SERVICE RESTART (no detail control statements needed)
INSTALL BACKOUT
    
```

### Detail Control Statements:

```

EXCLUDE PRODUCT=(CF781C)
INCLUDE PTF=(UD24500,UD34000)
    
```

## LIST



The LIST statement requests MSHP to print, on SYSLST, information from a service tape or service file.

### Logical Unit Assignments

Required:

#### SYS001

Work file used by MSHP.

#### SYS003

Work file used by MSHP.

#### SYS006

Service tape (not required for FROMDISK).

#### SYSLST

System printer.

Optional:

#### SYS002

Needed as work file if XREF is specified.

#### SYSxxx

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

none

Optional:

PTF (if COVER is specified)

### Description of Operands

#### SERVICETape

Specifies that information from a service file is to be printed.

#### FROMTape

Specifies that information from a service tape is to be printed.

#### FROMDisk

Specifies that information from the service file on disk is to be printed. The file must be defined with the following // DLBL statement:

```
// DLBL IJSYSPF,'PTF.FILE',,VSAM,CAT=VSESPUC
```

#### DOCument

Specifies printing of the service tape documentation, which contains

information on how to apply corrective and preventive service from the service tape. The DOCUMENT operand is not supported in combination with FROMDISK.

**NODOCument**

Suppresses the DOCUMENT function.

**XREF**

Specifies printing of the cross-reference list of all PTFs and APARs shipped on the service tape.

**NOXREF**

Suppresses the XREF option.

**COVer**

Specifies printing of the cover letters of those PTFs that are listed on an associated PTF detail control statement. If no PTF statement is given, the cover letters of all PTFs on the service tape are printed. The following is printed for all requested PTFs:

- Job control statements (including comments)
- MSHP control statements
- Librarian commands
- Linkage editor control statements

**NOCOVer**

Suppresses the COVER function.

**CONTInuous**

Specifies that the cover letters of the PTFs are to be printed without starting a new page for each PTF.

**SEParate**

Causes a new page to be started for each PTF cover letter that is to be printed.

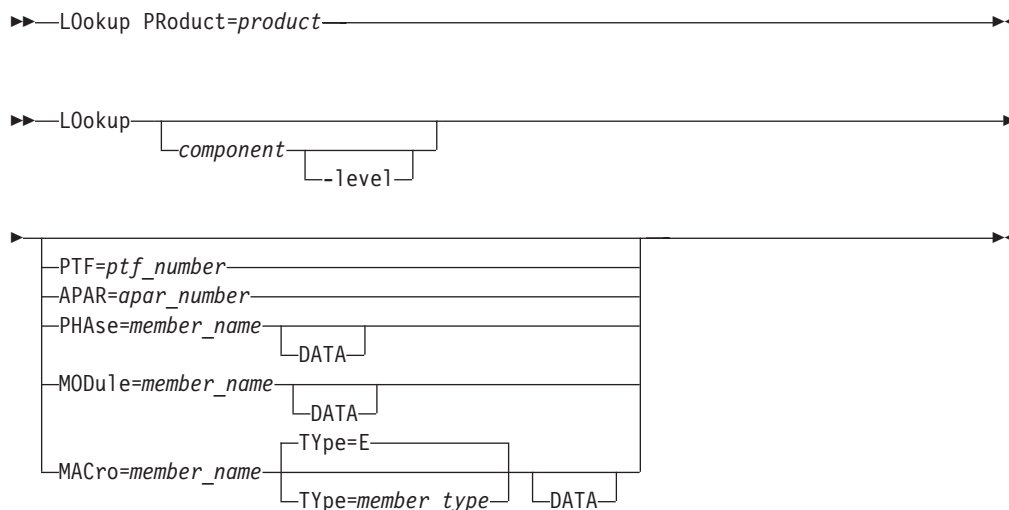
**Example:**

```
LIST SERVICETAPE COVER SEPARATE (default FROMTAPE is assumed)
LIST SERVICE FROMDISK
```

Detail Control Statement:

```
PTF=(UD34200,UD34201,UD34202)
```

## LOOKUP



The LOOKUP statement requests MSHP to display, on SYSLOG, selected information from the system history file.

### Logical Unit Assignments

Optional:

**SYSxxx**

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

none

Optional:

DEFINE HISTORY

### Description of Operands

#### **PProduct=product**

Indicates that the following information is to be displayed for the specified product-id:

- Date of installation.
- Requirements to be met for installation.
- Component(s) contained in the product.
- Comments, if any.

#### **component[-level]**

Specifies the component for which information is to be displayed. If level is omitted and more than one level of the component is installed, all levels of the component information will be displayed. If component is specified without any further operands, the following history file information is displayed:

- Component identifier plus release level.
- Latest service: number of the most recently applied PTF and its application date, or NO PTF applied.
- Latest APAR or latest fix application: number of the most recently applied local or APAR fix and its application date.



- Invalidated APARs: a list of local and/or APAR fixes that have been invalidated by the application of a PTF.
- Incomplete APARs: a list of local and/or APAR fixes whose application is incomplete.

**PTF=ptf-number**

Indicates that, for the given ptf-number, the following history file information is to be displayed:

- PTF number.
- Applied / Not applied / Revoked.
- Date of application (if applied).
- Superseded by / Not superseded.
- Affected component.
- Resolved APAR(s).
- Affected phases, modules, or macros.
- Prerequisites.
- Involved link-edits.
- Comments, if any, included in the PTF.

**APAR=apar-number**

Indicates that, for the given apar-number, the following history file information is to be displayed:

- APAR number (also for local fixes).
- Fixed / Not fixed by PTF / Local fix (if fixed).
- Date of correction (if fixed).
- Affected component.
- Affected phases/modules; if locally fixed and fix is recorded: alterations.
- Affected macros; if locally fixed and fix is recorded:
  - Insertions
  - Deletions
  - Replacements
- Comments, if any, included in the APAR.

**PHase=member-name**

Indicates that, for the given phase-name, the following history file information is to be displayed:

- Phase name.
- Not affected / Affected by PTF.
- Date when affecting PTF was applied by local fix.
- Date when local/APAR fix was made; if local fix was recorded: alterations.

**MODule=member-name**

Indicates that, for the given module-name, the following history file information is to be displayed:

- Module name.
- Not affected / Affected by PTF.
- Date when affecting PTF was applied by local fix.
- Date when local/APAR fix was made; if local fix was recorded:
  - CSECT
  - Expansion
  - Alterations

## LOOKUP

### **MACro=member-name**

Indicates that, for the given macro-name, the following history file information is to be displayed:

- Macro name.
- Not affected / Affected by PTF.
- Date when affecting PTF was applied by local fix.
- Date when local/APAR fix was made; if local/APAR fix was recorded:
  - Insertions
  - Deletions
  - Replacements

### **TType=member-type**

This member-type can be one character only or PROC or HTML.

If the operand is omitted, type E is assumed.

### **DATA**

Specifies that the source data from which the phase/module/macro was generated (with TAILOR KEEPDATA) is to be displayed.

### **Examples:**

```
LOOKUP 5686-CF7-06-81C PTF=UD00001
LOOKUP PRODUCT=CF781C
LOOKUP PTF=UD34500
LOOKUP APAR=DY34200
LOOKUP PHASE=$A$SUP1
```

## MERGE HISTORY



The MERGE HISTORY statement requests MSHP to insert entries of one history file into another history file.

The sequence of the keywords SYSTEM and AUXILIARY defines the direction of the merge operation. The first keyword specifies the source history file, and the second the target history file. The two keywords must be specified adjacent to each other.

*Restriction:* Both the source and the target history files must reside on disk.

### Logical Unit Assignments

Required:

#### **SYSLST**

System printer.

#### **SYSyyy**

The device on which the auxiliary history file resides. This can be either SYS002 or the device specified in the UNIT operand of the DEFINE HISTORY AUXILIARY statement.

Optional:

#### **SYSxxx**

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

## MERGE HISTORY

### Related Detail Control Statements

Required:

none

Optional:

DEFINE

### Description of Operands

#### SYStem AUXiliary

Specifies that entries from the system history file are to be merged into an auxiliary history file.

#### AUXiliary SYStem

Specifies that entries from an auxiliary history file are to be merged into the system history file.

**Example:** MERGE HISTORY SYS AUX

## PATCH

▶▶—PATCH Sublibrary=*lib.sublib*—◀◀

The PATCH control statement allows you to change (patch) a phase stored in a sublibrary. MSHP does not record the change in the history file.

The phase you patch may or may not be under control of MSHP. If the phase is MSHP-controlled, MSHP issues a warning message at the console.

### Logical Unit Assignments

When entered at the console, none.

Required when entered from SYSIPT:

#### SYSLST

System printer

### Related Detail Control Statements

Required:

AFFECTS

Optional:

ALTER

SCAN

When the control statements are entered from SYSLOG, the following additional commands are supported:

? To list supported control statements.

#### CANCEL

To undo previously entered ALTER statements.

The AFFECTS statement must precede any optional detail control statements.

### Description of Operands

#### Sublibrary=*lib.sublib*

For *lib* in *lib.sublib*, supply the name of the library that is to be accessed.

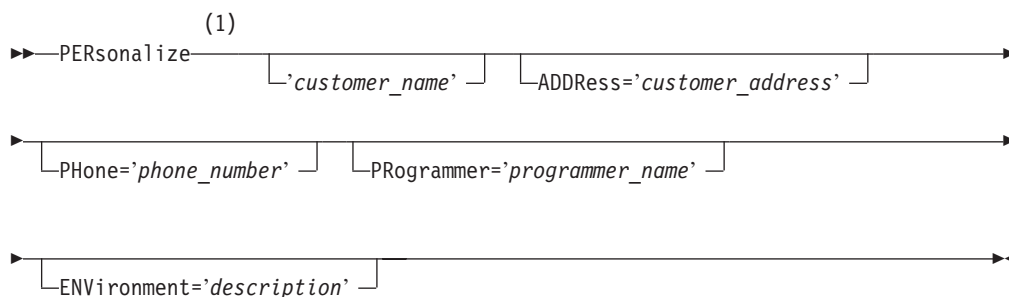
For *sublib* in *lib.sublib*, supply the name of the sublibrary in which the affected phase is stored.

#### Example:

Assuming that the phase to be changed resides in sublibrary PAYSUBL of library WEEKLIB, then your specification is:

```
PATCH  SUBLIBRARY=WEEKLIB.PAYSUBL
AFFECTS PHASE=MY PHASE
ALTER 094 47F0:4780
```

## PERSONALIZE



### Notes:

- 1 At least one operand must be specified.

The PERSONALIZE statement is used to identify a history file and relate it to a specific user.

### Restrictions:

- To personalize your system's history file, MSHP needs at least one operand.
- If the history file has not been personalized before, specification of customer-name and customer-address is mandatory.
- The first personalization of a history file changes the dates of all PTF entries to the date when the PERSONALIZE statement is given.

### Logical Unit Assignments

Required:

#### SYSLST

System printer.

#### SYSyyy

The device on which the auxiliary history file resides. This can be either SYS002 or the device specified in the UNIT operand of the DEFINE HISTORY AUXILIARY statement.

Optional:

#### SYSxxx

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

none

Optional:

DEFINE HISTORY

### Description of Operands

#### 'customer-name'

Specifies the user's name that is to be entered in the history file.

The name is a string of one to twenty characters, enclosed in quotes. If fewer than 20 characters are specified, the entry in the history file is padded with trailing blanks.

*Restriction:* If the history file has not been personalized before, customer-name must be specified.

**ADDRess='customer-address'**

Specifies the address that is to be entered in the history file.

The address is a string of 1 to 45 characters, enclosed in quotes. If fewer than 45 characters are specified, the string is padded with trailing blanks.

*Restriction:* If the history file has not been personalized before, customer-address must be specified.

**PHone='phone-number'**

Specifies the phone-number that is to be entered in the history file.

The phone number is a string of 1 to 17 characters, enclosed in quotes. If fewer than 17 characters are specified, the string is padded with trailing blanks.

A null string (two consecutive quotes) is accepted; it erases a previously specified number.

**PRogrammer='programmer-name'**

Specifies the programmer name that is to be entered in the history file.

The programmer name is a string of 1 to 24 characters, enclosed in quotes. If fewer than 24 characters are specified, the string is padded with trailing blanks.

A null string (two consecutive quotes) is accepted; it erases a previously specified name.

**ENVironment='description'**

Specifies any additional information (for example, the release level) that is to be entered in the history file.

The operand is a string of 1 to 62 characters, enclosed in quotes. If fewer than 62 characters are specified, the string is padded with trailing blanks.

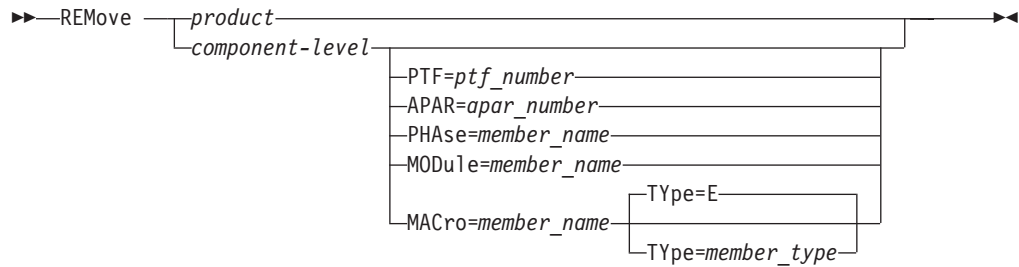
A null string (two consecutive quotes) is accepted; it erases a previously specified description.

**Example:**

```
PERSONALIZE 'M. Morris' -
ADDRESS='60 Water St., New York, N.Y.' -
PHONE='633 4537'
PROGRAMMER='JOHN'
ENVIRONMENT='REL2.1'
```

## REMOVE

### REMOVE



The REMOVE statement requests MSHP to erase entries from the system history file. The space of the removed history file entries is freed for future use.

**Note:** MSHP does not remove an APAR that was archived as a 'resolved' APAR in conjunction with a PTF.

### Logical Unit Assignments

Required:

**SYSLST**

System printer.

Optional:

**SYSxxx**

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

none

Optional:

DEFINE HISTORY

### Description of Operands

**product**

Indicates that the entry for the specified product is to be removed from the history file.

**Note:** Removing a product removes all COMPRISES-list(s) of this product but not the comprised component(s) itself.

**component-level**

Indicates that the entry for the specified component is to be removed (if no further operand follows).

**Note:** If a component is removed, the COMPRISES-list for this component is also removed from the corresponding product record in the history file. This implies that if all components comprised in a product were removed, the product record itself is removed too. An ARCHIVE of a REMOVED component cannot re-construct the COMPRISES list for a component of a product.

If followed by another operand, 'component' indicates the component to which the specified PTF, APAR, or member-name refers.



**PTF=ptf-number**

Indicates that the entries associated with the specified PTF number are to be removed.

**APAR=apar-number**

Indicates that the entry for the indicated APAR/local fix is to be removed.

**PHase=member-name**

Indicates that the entry for the specified phase name is to be removed. Only those entries can be removed which have been generated (see "GENERATE" on page 431 and "TAILOR" on page 416).

**MODule=member-name**

Indicates that the entry for the specified module name is to be removed. Only those entries can be removed which have been generated (see "GENERATE" on page 431 and "TAILOR" on page 416).

**MACro=member-name**

Indicates that the entry for the specified macro name is to be removed. Only those entries can be removed which have been generated (see "GENERATE" on page 431 and "TAILOR" on page 416).

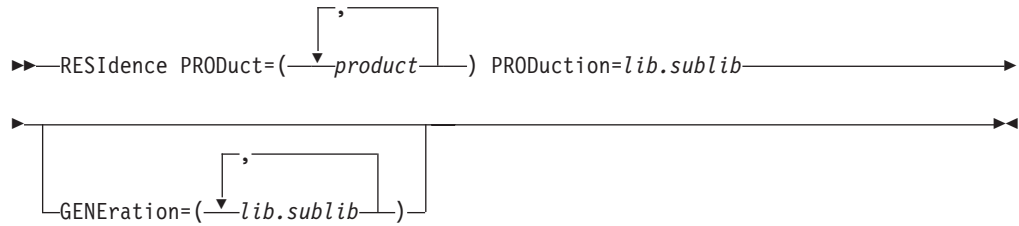
**TYpe=member-type**

This member-type can be one character only. Entries for members of type PROC or HTML cannot be removed.

If the operand is omitted, type E is assumed.

**Example:** REMOVE 5686-CF7-06-81C PTF=UD12345

## RESIDENCE



The RESIDENCE statement defines the names of the production and generation sublibraries in which the named products are to reside. This information is recorded in the history file for any follow-on activities, such as service applications, tailoring, installation, or product backup.

Any sublibrary names previously recorded in the history file (via another RESIDENCE or an INSTALL statement) are erased.

### Logical Unit Assignments

Required:

#### SYSLST

System printer.

Optional:

#### SYSxxx

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

none

Optional:

DEFINE HISTORY SYSTEM

### Description of Operands

The operands PRODUct= and PRODUction= must be entered in the sequence as shown.

#### PRODUct=(product,...)

Specifies the name(s) of the product(s) whose residence is to be defined.

#### PRODUction=lib.sublib

Indicates that the production part of the product(s) is to reside in the specified sublibrary.

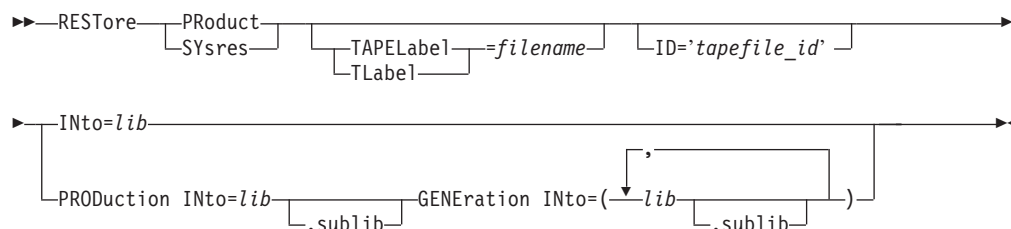
#### GENERation=(lib.sublib,...)

Indicates that the generation part of the product(s) is to reside in the specified sublibrary.

Example:

```
RESIDENCE PRODUCT=099360 PRODUCTION=PRD1.BASE
RESIDENCE PRODUCT=LM4E11 PRODUCTION=LIB01.PR$E11 -
GENERATION=LIB01.G1$E11
```

## RESTORE PRODUCT/SYSRES



The RESTORE statement is used to restore a complete shipment tape (production part, generation part, and shipment history file) onto disk; however, without any checks or updates of the system history file. The history file included in IBM's shipment tape is restored into an auxiliary history file. If you want to use this history file, make sure that it is located in SAM space.

### Logical Unit Assignments

Required:

#### SYS006

Distribution tape.

#### SYSLST

System printer.

#### SYSyyy

The device on which the auxiliary history file resides. This can be either SYS002 or the device specified in the UNIT operand of the DEFINE HISTORY AUXILIARY statement.

Optional:

#### SYSxxx

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

none

Optional:

DEFINE HISTORY AUXILIARY

### Description of Operands

#### PProduct

Specifies that a non-SYSRES package is to be restored.

#### SYSres

Specifies that a SYSRES shipment package is to be restored.

#### TAPELabel | TLabel=filename

Specifies that the shipment tape contains standard labels. **filename** is the seven-character filename that is specified in the // TLBL statement for the shipment tape.

## RESTORE PRODUCT/SYSRES

### **INTo=lib**

Specifies, for RESTORE PRODUCT only, that both the production and the generation part of the shipment package are to be restored into the library denoted by lib.

**Note:** Cannot be used for RESTORE SYSRES, because the production part and the generation part must be restored into different target libraries.

### **PRODUCTION INTO=lib[.sublib]**

Specifies that the production part of the shipment package is to be restored to the named library (and sublibrary).

For RESTORE PRODUCT, the target library must exist (online), and label information must be available for it in the label area.

For RESTORE SYSRES, the name of the target library for the production part must be IJSYSRn, n being a digit from 1 to 9. (The name of the target library for the generation part must be different.) IJSYSRn will be created by MSHP if it does not exist.

If you omit 'sublib', MSHP takes the name of the shipment sublibrary as default.

### **GENERATION INTO=(lib[.sublib],...)**

Specifies that the generation part of the shipment package is to be restored to the named library or libraries.

For RESTORE PRODUCT, the target libraries must exist (online), and label information must be available in the label area.

For RESTORE SYSRES, the target library is created by MSHP if it does not exist (label information must be available in the label area).

If you omit 'sublib', MSHP takes the name of the shipment sublibrary as default.

### **ID='tapefile-id'**

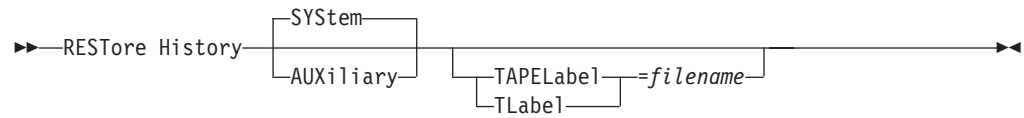
Indicates that the shipment tape is to be searched for the denoted tapefile-id, which was specified in the BACKUP statement. If the tape is not correctly positioned, it is scanned (forward only) for the specified tapefile-id and correctly positioned.

The tapefile-id can be 1 to 16 characters.

If you omit the operand, the tape is assumed to be correctly positioned.

**Example:** RESTORE PRODUCT INTO=PRODLIB

## RESTORE HISTORY



The RESTORE HISTORY statement requests MSHP to write a history file located on magnetic tape onto disk. If the tape containing the history file was not written using the BACKUP HISTORY statement, the tape must be positioned at the file containing the history file before you issue the RESTORE statement.

### Logical Unit Assignments

Required:

#### **SYS006**

The tape containing the history file.

#### **SYSLST**

System printer.

Required additionally for RESTORE HISTORY AUXILIARY:

#### **SYSyyy**

The device on which the auxiliary history file resides. This can be either SYS002 or the device specified in the UNIT operand of the DEFINE HISTORY AUXILIARY statement.

Optional:

#### **SYSxxx**

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Optional:

DEFINE HISTORY

### Description of Operands

#### **SYStem**

Specifies that a history file on tape is to be copied to the system history file (a disk file with the file name IJSYSHF).

#### **AUXiliary**

Specifies that a history file on tape is to be copied to the auxiliary history file (a disk file with the file name IJSYS02).

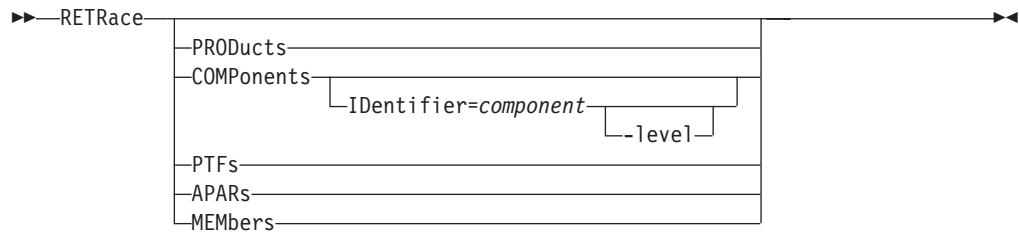
#### **TAPeLabel | TLabel=filename**

Specifies that the history file tape contains standard labels. **filename** is the seven-character filename that is specified in the // TLBL statement for the tape.

**Example:** RESTORE HISTORY AUX

## RETRACE

### RETRACE



The RETRACE statement requests MSHP to print information from the system history file on SYSLST. The listings produced are identified with the PERSONALIZE information (contained in the history file header record).

On a system running under IBM VSE/System Package Version 3, Release 1, listing includes the refresh level indication of the system.

*Defaults:* If RETRACE is specified without any keywords, MSHP writes, to SYSLST, a report on the system's current service level; this report contains:

1. A list of all products and components installed.
2. A combined list of all local fixes, sorted by APAR number, and of all applied and not superseded PTFs, sorted by PTF number.
3. An APAR cross-reference list. It lists, for all APARs that have been applied to the system (in APAR number sequence), whether a particular APAR was corrected locally (local fix) or whether it has been resolved by a PTF and, if so, by which PTF.
4. A member cross-reference list. It lists, for all sublibrary members that were affected by a PTF or local fix, the affecting PTF or APAR number. This listing is in alphabetical order by member name (without respect to member type).

### Logical Unit Assignments

Required:

#### SYSLST

System printer.

Optional:

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

none

Optional:

DEFine HISTORY

### Description of Operands

#### PRODUCTs

Requests MSHP to print a list of all products installed, together with the following information:

- Date of installation (or personalization)
- Component(s) contained in the product
- Comments, if any

**COMPONENTS**

Requests MSHP to print a list of all component records from the history file. The printout includes (for each component):

- Component identifier plus release level
- Date of installation (or personalization)
- All PTFs applied to the particular component (in PTF number sequence)
- All APARs and local fixes applied to the particular component (in APAR number sequence)
- All generated members (for the TAILOR function)

**COMPONENTS IDENTIFIER=component[-level]**

Requests MSHP to print information for the specified component only. If level is omitted, MSHP prints information for all installed levels of the component.

**PTFS**

Specifies that all applied PTFs are to be listed (in PTF number sequence). For each PTF the following information is printed:

- PTF number
- Whether or not the PTF was revoked
- Component to which the PTF applies
- Affected members
- Resolved APARs
- Prerequisites, corequisites, and also negative prerequisites
- PTFs which this PTF supersedes
- The PTF that supersedes this PTF, and the date of application of the superseding PTF

**APARs**

Specifies that all APARs are to be listed (in APAR number sequence) which were corrected by a PTF or local/APAR fix. For each APAR the following information is printed:

- APAR number
- Component to which the APAR applies
- PTF number, if the APAR has been resolved by a PTF
- Date of correction
- If locally corrected:
  - Affected modules
  - What the fix consisted of

**MEMBERS**

Specifies that all phases, modules, and macros that are affected by a PTF or local fix are to be listed. For each member the following information is printed:

- Member name
- Component to which the member belongs
- Date of PTF or local fix application
- PTF number, if affected by a PTF
- APAR number; if locally corrected, also what the fix consisted of

**Note:** Since RETRACE MEMBERS does not indicate whether an APAR, PTF, or component is incorrect or incomplete, use RETRACE APARS|PTFS|COMPONENTS instead.

**Examples:**

```
RETRACE PRODUCTS
RETRACE COMP ID=5686-CF7-06-81C
RETRACE PTFS
```

## REVOKE

►►—REVOKE *component* [*-level*] :*ptf\_number*—►►

The REVOKE statement initiates a backout PTF job that contains the phases, modules, and macros as they were before the named PTF was installed. This backout PTF job (with the initial REVOKE statement) is generated by the APPLY or INSTALL SERVICE statements if REVOKABLE was specified.

By bringing the backout PTF back onto the system (with the INSTALL BACKOUT statement), MSHP restores the system to the status that existed before the original PTF was installed. MSHP also flags the history file entry for the PTF as revoked.

A PTF cannot be revoked if it is a prerequisite for another PTF that has not been revoked previously.

### Logical Unit Assignments

Required:

#### SYSLNK

Linkage editor input file.

#### SYS001

Linkage editor work file.

#### SYSLST

System printer.

Optional:

#### SYSxxx

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

DATA

Optional:

none

### Description of Operands

#### **component**[-level]

Identifies the component for which the backout PTF was generated.

If level is not specified and only one level of the specified component is installed, the REVOKE job is applied to this one. If two or more levels are installed, MSHP informs you which levels are installed and asks you to which one the REVOKE job applies.

#### **ptf-number**

Identifies the PTF that is to be revoked. The number to be specified is that of the originally applied PTF (the one that proved to be unsatisfactory). The PTF is flagged as 'revoked' in the system history file.

**Example:** REVOKE 5686-CF7-06-81C : UD00001



## SELECT

```

▶▶—SElect GENFile COMPonent=component—————▶▶
                                     └-level┘

```

The SELECT statement identifies the generation file, from which individual phases, modules, or macros can be regenerated (with the GENERATE detail control statement) after a service application.

The generation file is a set of MSHP tailor jobs, each of which must be preceded by a // JOB statement and followed by a /& statement. The generation file may reside on tape, disk, or diskette. The records of the file can be 80 or 81 bytes long. The generation file can also be a card deck, in which case it must be terminated by a /\* card, immediately followed by the last /& card.

### Logical Unit Assignments

Required:

**SYS005**

Generation file.

**SYSLNK**

Linkage editor input file.

**SYS001**

Linkage editor work file.

**SYSLST**

System printer.

Optional:

**SYSxxx**

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

### Related Detail Control Statements

Required:

GENERATE

Optional:

DEFINE HISTORY

### Description of Operands

**GENFile**

Indicates the generation file.

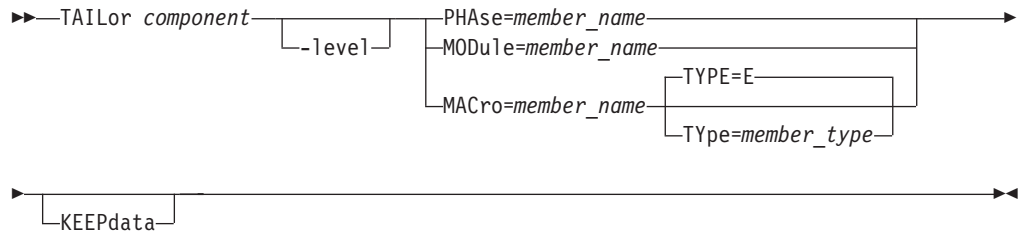
**COMPonent=*component*[-level]**

Identifies the component to which the members to be regenerated belong.

The level indication must be the same as that of the corresponding TAILOR job. If level was not specified during tailoring or retailoring, and multiple levels of the affected component are installed, MSHP asks you for which level the generation is to be done.

**Example:** SELECT GENFILE COMPONENT=5686-CF7-06-81C

## TAILOR



This statement is supported for compatibility reasons and refers to the former **DOS/VSE Assembler only**. (The DOS/VSE Assembler has been replaced by the High Level Assembler for VSE). The TAILOR statement (together with the EXECUTE detail control statement) can be used to generate (tailor) sublibrary members of components that are shipped in source-macro format and that have to be assembled and link-edited according to the specific needs of your installation (for example, supervisor macros).

**Note:** Members of type PROC or HTML cannot be generated.

If later a PTF is installed to a generated sublibrary member, you may have to regenerate (retailor) the member (also with the TAILOR statement) in order to make the applied fix active for the system. Note, however, that it is not possible to retailor a sublibrary member directly after having tailored it, that is, without a prior service application. Also, if a phase is affected directly by a PTF, that phase cannot be retailored at all.

### Logical Unit Assignments

Required:

**SYSLNK**

Linkage editor input file.

**SYS001**

Linkage editor/assembler work file.

**SYS002,**

**SYS003**

Assembler work files.

**SYSLST**

System printer.

Optional:

**SYS004**

Work file needed by MSHP if MODULE= or MACRO= is specified.

**SYSxxx**

Required if the device on which the system history file resides is specified in the UNIT operand of the DEFINE HISTORY SYSTEM statement.

## Related Detail Control Statements

Required:

EXECUTE

Optional:

RESOLVES

DEFINE

## Description of Operands

### **component**[-level]

Identifies the component containing the macro, module, or phase to be tailored.

If level is omitted and only one level of the affected component is installed, MSHP generates the sublibrary member for this one; if two or more levels are installed, MSHP informs you which levels are installed and asks you for which one you want to generate the member.

### **PHASe=member-name**

Specifies the name of the phase to be generated. (See Note, below.) For retailoring, generic names such as DFH\* are allowed.

### **MODUle=member-name**

Specifies the name of the module to be generated. (See Note, below.) For retailoring, generic names such as DFH\* are allowed.

### **MACRO=member-name**

Specifies the name of the macro (definition) to be assembled. (See Note, below.) For retailoring, generic names such as DFH\* are allowed.

**Note:** The operands PHASE=, MODULE=, or MACRO= do *not* generate any PHASE or CATALOG statements. You have to include these statements after the EXECUTE detail control statement. MSHP uses the information to:

- Check if a specified member already exists in the history file.
- Compare the PHASE, MODULE, or MACRO operands with the corresponding PHASE or CATALOG statements given.
- Enter the member name into the history file.

### **TYpe=member-type**

This member-type can be one character only.

If the operand is omitted, type E is assumed.

### **KEEPdata**

Specifies that the source code processed by the invoked control program(s) is to be stored in the system history file. MSHP uses that code if retailoring is to be done later on. Instead of resubmitting the original tailor job, it is sufficient to invoke MSHP and initiate, for example, the function

```
TAILOR component PHASE=member-name
```

without an EXECUTE detail control statement.

MSHP fetches, from the system history file, all information and data belonging to the generated phase and reassembles (and recatalogs) it.

For those tailor jobs that are too large to be kept in the history file, do not specify KEEPDATA. Instead, create a sequential generation file (with the filename GENFILE) and put the original tailor job(s) into it. To create this file,

## TAILOR

you can use a program of your own or an IBM-supplied program such as OBJMAINT or DITTO (Data Interfile Transfer, Testing, and Operations utility).

Each individual tailor job on the generation file must be preceded by a // JOB statement and followed by a /& statement. The generation file must be assigned to SYS005; it can be on tape, disk, or diskette, and the records must be 80 or 81 bytes long. The generation file can also be a card deck, in which case the last card must be a /\* card, immediately followed by the last /& card.

You can then retailer individual members from the generation file by invoking MSHP with the SELECT GENFILE function control statement and a GENERATE detail control statement for each member to be retailed.

**Example:** TAILOR 5686-CF7-06-81C PHASE=\$A\$SUP8 KEEPDATA

## UNDO

►► UNdo *component* [*-level*] :*apar\_number* ◀◀

The UNDO statement is used to re-establish the status of a sublibrary member as it existed before a local or APAR fix was applied with the CORRECT...REVOKABLE statement.

*Restriction:* If a phase has been expanded by a local or APAR fix, this expansion cannot be removed. The phase remains expanded.

For phases and modules, MSHP may be invoked with an UNDO statement that refers (by component and apar-number) to the correction as specified in the CORRECT statement; MSHP uses the information to remove the correction from the respective library and the system history file. For macros, the UNDO statement is included in the job created (on SYSPCH) by CORRECT...REVOKABLE.

### Logical Unit Assignments

Same as for CORRECT.

### Related Detail Control Statements

Required:

none

Optional:

DEFINE HISTORY SYSTEM

DATA

### Description of Operands

#### **component**[*-level*]

Specifies the component from which the local or APAR fix (initiated by CORRECT) is to be removed. If level is omitted and only one level of the component is installed, MSHP removes the fix from this one. If two or more levels are installed, MSHP informs you which levels are installed and prompts you for the level of the applicable component.

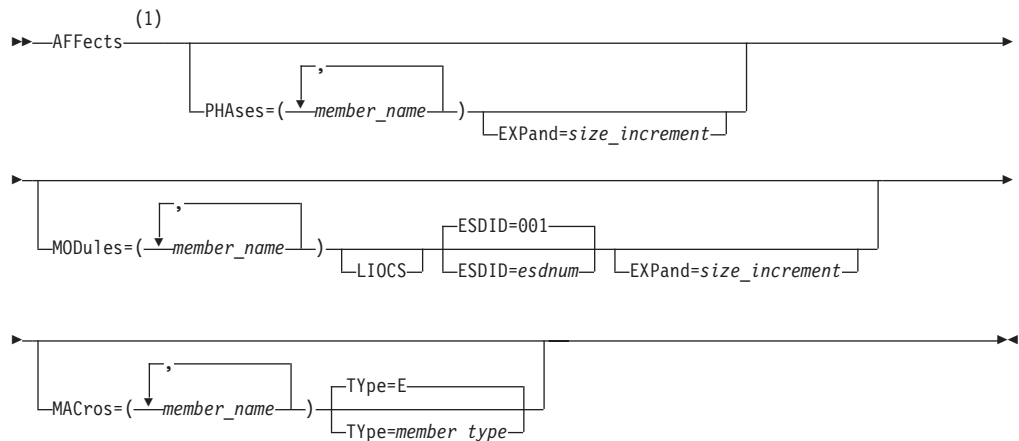
#### **apar-number**

Specifies the local or APAR fix (initiated by CORRECT) that is to be removed.

**Example:** UNDO 5686-CF7-06-81C : DY19227

## Detail Control Statements

### AFFECTS



#### Notes:

- 1 At least one operand must be specified

The AFFECTS statement identifies the phases, modules, and macros that are affected by a PTF or local fix application.

#### Restrictions:

One AFFECTS statement may not refer to more than a total of 100 phases, modules, and macros.

If AFFECTS is used as a detail control statement to CORRECT, or when archiving a local/APAR fix and the fix information itself, only one phase, module or macro may be specified.

### Description of Operands

#### PHAses=(member-name,...)

Specifies the affected phase(s).

#### EXPand=size-increment

Indicates that the specified phase or module (see below) is to be made larger by the number of bytes specified in size-increment, so that fix code can be added at the end of the phase or module. Size-increment is a decimal number of one to six digits.

EXPAND may be specified only when applying a local/APAR fix (with CORRECT) or archiving a local/APAR fix.

#### MODules=(member-name,...)

Specifies the affected module(s).

#### LIOCS

Indicates that a LIOCS module is affected by the PTF. However, only the macro needed to generate the module is distributed in the PTF, but not the affected module itself.

#### ESDId=esd-number

Indicates that a change applies to the specified ESD.

*Default:* If not specified, ESDID=001 is assumed, that is, the first ESD.

*Restriction:* ESDID may be specified only when correcting a component (CORRECT) or archiving (ARCHIVE) a local/APAR fix.

For esd-number specify one to three hexadecimal digits. If less than three digits are specified, the number is padded with leading zeros.

**MACros=(member-name,...)**

Specifies the affected macro(s).

**Type=member-type**

This member-type can be one character only or PROC or HTML.

If the operand is omitted, type E is assumed.

**Example:** AFFECTS MACRO=GENAB TYPE=A

**Note:** For compatibility reasons, the operand CSect=csect-number is still accepted (for ESDid).

## ALTER

▶—ALTER *address old\_text:new\_text*—▶

The ALTER statement identifies the modifications that are to be made to a phase or module. This includes verification of the alteration for phases and, optionally, for modules.

### Description of Operands

#### address

Specifies the (relative) address where the new-text is to begin to replace the old-text.

Address is a string of one to six hexadecimal digits. Leading zeros may be omitted.

#### old-text

Specifies the text that is to be replaced.

MSHP checks the text in the phase or module at the specified address whether it is identical with the old text; replacement by new text takes place only if the text is identical.

Old-text *must* be specified when modifying a phase; it *may* be specified when modifying a module. When a module or phase has been expanded, old-text must not extend from the module or phase data into the expanded area. Instead, a separate ALTER statement is needed for the expanded area.

If the phase is expanded and the old text is in the expanded area, specify a pair of hexadecimal 0's for each byte in 'old-text'.

Old-text can be in one of the following formats:

- An even-numbered string of 2 to 32 hexadecimal digits, where one pair of hexadecimal digits describes one byte in the phase.  
The same applies to modules; however, the specification must be a multiple of 4 digits.
- A string of one to sixteen characters, enclosed in quotes, where each character represents one byte in the phase.  
The same applies to modules; however, the specification must be an even number of bytes.
- A repetition factor, which is a decimal value that indicates how often the associated string of hexadecimal digits occurs in the resulting text string. This string must not exceed 32 hexadecimal digits.  
The specified repetition factor must be 2 or higher; it precedes, without intervening blanks, the associated string and must be enclosed in slashes.  
For example:  
/16/FF (means sixteen FF's)

#### new-text

Specifies the text that is to replace the text at the specified address (see above).

New text can be in any of the formats described under old-text (see above).

If old-text is specified, new-text must have the same length (in bytes).

If new-text is specified without old-text, the colon must be specified at the beginning of the new-text line.

**Example:** ALTER 2034 47F0F000 : 47F0F800



## COMPATIBLE

►► COMPATible WITH=(*product*) ◀◀



The COMPATIBLE statement is used to indicate to MSHP at installation time those products that are compatible with the shipped product(s).

Compatible products are usually based on the same base products, contain the same components as the shipped products. Compatible products may run concurrently with each other, and may also be stored in the same sublibrary.

### Description of Operands

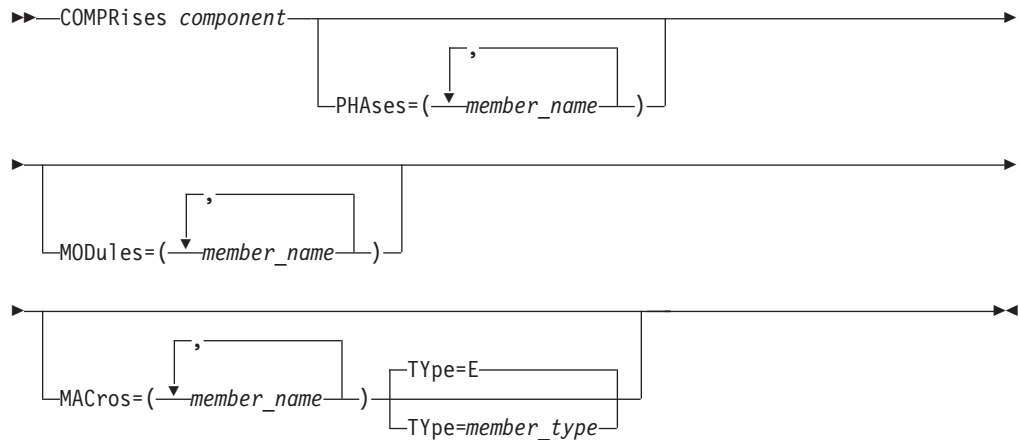
**WITH=(product,...)**

Specifies the name(s) of the compatible product(s).

**Example:** COMPATIBLE WITH=CF781C

## COMPRISES

### COMPRISES



The COMPRISES statement is used to specify the component(s) comprised in the shipped product and the sublibrary members that make up the component(s). The information is entered in the system history file. A separate COMPRISES statement must be issued for each component contained in the shipped product.

*Restriction:* One COMPRISES statement may not refer to more than a total of 100 phases, modules, and macros. Use multiple COMPRISES statements for the same component, if necessary.

### Description of Operands

#### component

Specifies the component comprised in the shipped product.

#### PHAses=(member-name,...)

Specifies the phases of the named component.

#### MODules=(member-name,...)

Specifies the modules of the component.

#### MACros=(member-name,...)

Specifies the macros of the component.

#### TYpe=member-type

This member-type can be one character only or PROC or HTML.

If the operand is omitted, type E is assumed.

#### Example:

```
COMPRISES 5686-066-02 -
          PHASES=PHASA* -
          MODULES=MODA* -
          MACROS=MACA*
COMPRISES 5686-066-03 -
          PHASES=PHASB -
          MODULES=MODB*
```

## DATA

▶▶—DATA—◀◀

The DATA statement in conjunction with /\$ delimits input that is to be passed by MSHP to the linkage editor or the librarian.

### *Restrictions:*

- A DATA statement (with its corresponding terminating delimiter /\$) may be followed only by another DATA statement, not by any other detail control statement.
- The end-of-data indicator (/ \$) is valid only when input is entered via SYSIPT. Substitute this delimiter by hitting END/ENTER if input is entered from the console.
- Input for the linkage editor must not contain 'named INCLUDE' statements; however, this is not checked by MSHP.

Linking from a link-book (where link-book is an object module that contains LNKEDT control statements) must be requested with the MSHP statement INVOLVES.

The sequence of delimiters and input is as follows:

### **DATA**

The initiating delimiter

### **Input**

Linkage editor or librarian statements

/\$ The terminating delimiter, which must be in columns one and two in an input line, followed by 70 blanks.

### **where:**

Input refers to data on SYSIPT after // EXEC MSHP has been read from SYSRDR.

MSHP checks the first line after the DATA statement. If this is a linkage editor control statement, all input beginning with the statement and up to, but excluding, the next /\$ or /\* line, is passed unaltered and unchecked to the linkage editor. If it is a librarian control statement, MSHP passes the input in the same way to the librarian program.

The linkage editor control statements checked for are:

ACTION  
ENTRY  
INCLUDE  
PHASE

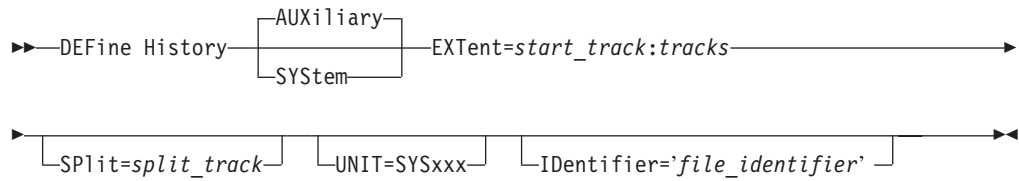
The librarian control statement checked for is:

CATALOG

MSHP internally converts any old MAINT CATALR and CATALS statements into the CATALOG statement.

**Example:** DATA

## DEFINE HISTORY



The DEFINE HISTORY statement is used to create extent definitions for a history file in the user label area of the partition in which MSHP is executed.

*Restrictions:*

- If you use IBM supplied standard labels or if your own standard label set contains DLBL and EXTENT statements for the system history file (filename IJSYSHF), do not use DEFINE HISTORY SYSTEM in any MSHP job accessing the system history file.
- A DEFINE statement, if used, must immediately follow the applicable function control statement; it may not be placed at the end of several functions or at the end of the job stream.

### Description of Operands

#### AUXiliary

Specifies that an auxiliary history (work) file is to be defined. This is the default, if neither AUXiliary nor SYStem is specified.

The auxiliary history file is maintained under the filename IJSYS02 on the default logical unit SYS002. MSHP normally uses this file as a history work file. As such, the permanent DLBL and EXTENT definition that most systems contain for the IJSYS02 work file is sufficient. The DEFINE HISTORY AUXILIARY statement allows you to explicitly define a temporary auxiliary history file (in the user label area) on the logical unit indicated in the UNIT operand.

**Note:** When you explicitly define the workfile under VSE/VSAM space management, a RECSIZE of 2000 is required.

#### SYStem

Specifies that the system history file is to be defined.

The system history file is part of the IBM distributed system and is maintained under the filename IJSYSHF. To access the file, MSHP uses the IBM set default logical unit SYSREC. However, you can use any programmer logical unit to refer to the file if you place it on a volume other than that of SYSREC.

The history file should be permanently defined. If it is to be on the SYSREC volume, supply the following statements:

```
// DLBL IJSYSHF, 'VSE.SYSTEM.HISTORY.FILE', 99/365, SD
// EXTENT SYSREC, , 1, 0, start-address, number-of-tracks/blocks
```

With the DEFINE HISTORY SYSTEM statement you can define a temporary system history file on the logical unit specified in the UNIT operand. This definition is, of course, valid only for the duration of the applicable MSHP job.

Since the system history file normally contains all the status information of the system, you should always keep a backup copy of it.

### **EXTent=start-track:tracks**

Specifies the extent information for the history file.

Start-track specifies the sequential number of the track (relative to zero) where the extent is to begin. For FBA devices, start-track indicates the block number at which the extent is to start.

Tracks specifies the number of tracks (or FBA blocks) to be allocated to the history file.

For the number of tracks or blocks required on the various types of disk volumes, see “z/VSE Disk Layouts” in the manual *z/VSE Installation*.

### **SPlit=split-track**

Specifies, for CKD devices, which track is the last one in each cylinder to be allocated to the history file. (The first cylinder occupied by the file is the one in which the “start-track” lies, and the last cylinder is determined by the number of tracks specified.)

Split-track is a two-digit decimal integer equal to the number of tracks per cylinder minus one.

### **UNIT=SYSxxx:**

Specifies the logical unit (other than SYSREC) on which the history file is to reside.

*Defaults:* If not specified, MSHP takes the following defaults:

- For a system history file: SYSREC
- For an auxiliary history file: SYS002

### **IDentifier='file-identifier'**

Specifies the history file identification that is to be entered in the VTOC.

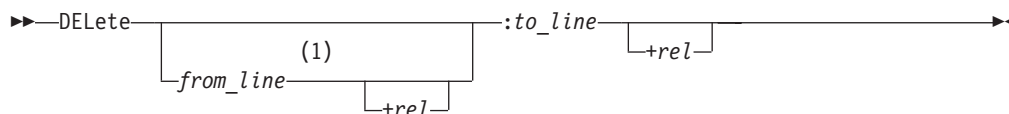
The file-identifier is a string, enclosed in quotes, of one to 44 alphanumeric characters. *Defaults:* If the operand is not specified, MSHP takes the following defaults:

- For an auxiliary history file:  
‘VSE.AUXILIARY.HISTORY.FILE’
- For the system history file:  
‘VSE.SYSTEM.HISTORY.FILE’

**Example:** DEFINE HIST EXTENT=19:96

## DELETE

### DELETE



#### Notes:

- 1 Default: If the operand is omitted, `from_line` is assumed to be equal to the `to_line` value.

The DELETE statement indicates the lines to be deleted from a macro (definition) when applying a local/APAR fix.

**Note:** The DELETE statement cannot be used for members of type PROC or HTML.

#### Description of Operands

##### from-line

Specifies the line-number (in columns 73 through 78 in the macro) where deletion begins. The from-line is the first line to be deleted.

*Default:* If omitted, from-line is assumed to be equal to the to-line value. This means that only the line designated by to-line is deleted.

##### +rel

Identifies the position of the statement relative to the from-line number.

rel is an integer of one or two digits. It applies to E- or F-type macros only.

##### to-line

Identifies the last line of the lines to be deleted. The value of to-line must be equal to or greater than the value given in from-line.

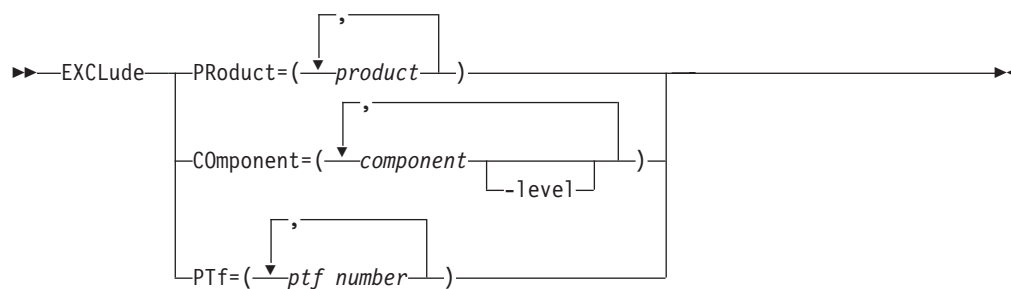
##### +rel

Identifies the position of the statement relative to the to-line number.

rel is an integer of one or two digits. It applies to E- or F-type macros only.

**Example:** DELETE 000380:000400

## EXCLUDE



The EXCLUDE statement is used to exclude specific products, components or PTFs from a service application (with the INSTALL SERVICE statement). This implicitly includes service for all other products, components, or PTFs shipped on the tape.

### Description of Operands

#### PRoduct=(product,...)

Specifies the product(s) that are not to be serviced.

#### COmponent=(component[-level],...):

Specifies the component(s) that are not to be serviced. If level is not specified, MSHP excludes all levels of the component.

#### PTf=(ptf-number,...)

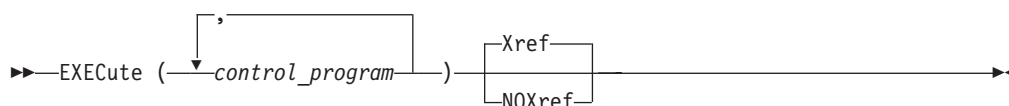
Lists the PTFs that are not to be installed.

#### Examples:

```
EXCLUDE PRODUCT=A048TP
EXCLUDE COMPONENT=5686-CF7-06-81C
```

## EXECUTE

### EXECUTE



The EXECUTE statement is used to indicate which system programs (assembler, librarian or linkage editor) are to be called in which order to process the data submitted with the TAILOR statement. The data to be processed must immediately follow the EXECUTE statement and be terminated by /\$.

#### Description of Operands

##### (control-program,...)

MSHP calls the specified system program(s) in the submitted order to process the data which immediately follows the EXECUTE statement (and is terminated by /\$).

If two programs are specified, the output of the first program is taken as input to the second without any modification.

Any mismatch between the program and the data (for example, an object deck as input for ASSEMBLY) is not checked by MSHP, but results in an error situation diagnosed by the called program.

The following programs or program combinations can be specified:

```
EXEC ASSEMBLY,LNKEDT
EXEC ASSEMBLY,LIBR
EXEC LNKEDT
EXEC LIBR
EXEC ASSEMBLY EXEC LNKEDT
EXEC ASSEMBLY EXEC LIBR
```

MSHP internally converts any reference to the old MAINT program into a reference to the new LIBR program.

#### CAUTION:

**ASSEMBLY is valid for the DOS/VSE Assembler only.**

#### Xref

Specifies that the cross-reference list of included macros as given by the ASSEMBLY program is to be recorded in the history file.

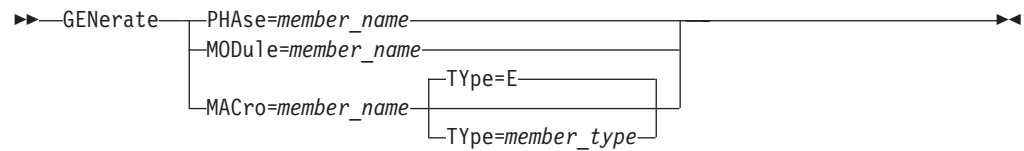
#### NOXref

Specifies that the cross-reference list of included macros is not to be recorded in the history file.

**Example:** EXECUTE (ASSEMBLY,LNKEDT)



## GENERATE



The GENERATE statement is used as a detail control statement to the SELECT statement to regenerate (retailor) individual phases, modules, or macros from the generation file.

### Description of Operands

#### **PHase=member-name**

Indicates to MSHP the name of the phase that is to be regenerated.

#### **MODule=member-name**

Indicates to MSHP the name of the module that is to be regenerated.

#### **MACro=member-name**

Indicates to MSHP the name of the macro that is to be regenerated.

#### **TYpe=member-type**

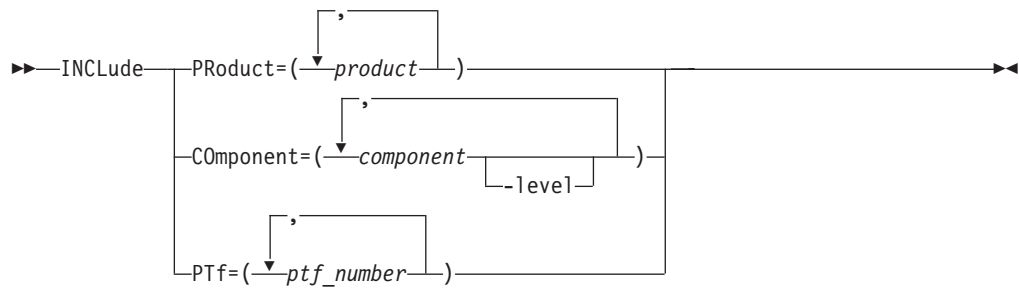
This member-type can be one character only. Members of type PROC or HTML cannot be generated.

If the operand is omitted, type E is assumed.

**Example:** GENERATE MACRO=BMS030 TYPE=A

## INCLUDE

### INCLUDE



The INCLUDE statement is used to indicate to MSHP that only the named products, components, or PTFs are to be included in a service application (with INSTALL SERVICE). This implicitly excludes service for all other products, components, or PTFs shipped on the service tape, except prerequisites and corequisites.

#### Description of Operands

##### **PProduct=(product,...)**

Specifies the product(s) to which service is to be applied.

##### **CComponent=(component[-level],...)**

Specifies the component(s) to which service is to be applied.

If level is omitted, all levels of the component are serviced.

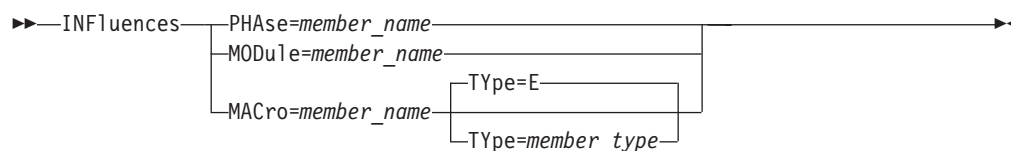
##### **PTf=(ptf-number,...)**

Lists the PTFs that are to be applied.

##### **Examples:**

```
INCLUDE PRODUCT=CF781C
INCLUDE COMPONENT=5686-CF7-06-81C
```

## INFLUENCES



The INFLUENCES statement identifies which generated phases, modules, or macros of the serviced component are affected by a PTF or local/APAR fix and have to be regenerated.

*Restriction:* One INFLUENCES statement may not refer to more than a total of 100 phases, modules, and/or macros.

### Description of Operands

#### PHase=(member-name,...)

Names the phases to be regenerated.

#### MODule=(member-name,...)

Names the modules to be regenerated.

#### MACro=(member-name,...)

Names the macros to be regenerated.

#### TYpe=member-type

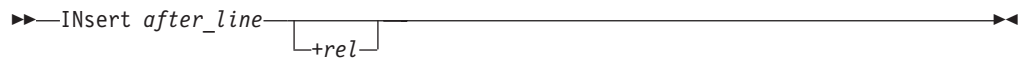
This member-type can be one character only. Members of type PROC or HTML cannot be generated.

If the operand is omitted, type E is assumed.

**Example:** INFLUENCES PHASE=DFHF\*CP\*

## INSERT

### INSERT



The INSERT statement identifies where, in a source book (macro), additions are to be made when archiving a local/APAR fix or when initiating a local or APAR fix by means of the CORRECT statement. The statement further serves as the initiating delimiter A /\$ (on SYSIPT) or a blank line (at the console) is the terminating delimiter for the input line to be inserted.

**Note:** The INSERT statement cannot be used for members of type PROC or HTML.

#### Description of Operands

##### after-line

Specifies the line number in the macro (in columns 73 through 78) after which the source input (following the INSERT statement up to the next /\$) is to be inserted.

after-line is an integer of one to six digits. If fewer than six digits are coded, leading zeros are supplied.

##### +rel

Specifies the position of the source input relative to the after-line number.

rel is an integer of one or two digits.

*Restriction:* rel applies to E- or F-type macros only.

**Example:** INSERT 7100

## INVOLVES

▶▶—INVolves LINK=()—▶▶

The INVOLVES statement explicitly requests link-editing to be performed when installing an archived product, or when applying PTFs from a service tape.

As a detail control statement to APPLY, INCORPORATE, and CORRECT, it indicates that, as the final step of the particular function, a link-edit run must be performed.

### Description of Operands

#### LINK=(link-book,...)

Link-book specifies the name of a module that is to be included in the link-edit step.

Link-book is a string of one to eight characters, the first one of which must *not* be an asterisk.

If you specify several link-books, the linkage editor includes the named modules in the same sequence as they occur in the list. You can specify up to 100 link-books on a maximum of 15 lines. The linkage editor is called for each link-book specified.

**Example:** INVOLVES LINK=IJWIND

OR

OR

►►—OR—◄◄

The OR statement initiates a set of alternative REQUIRES statements that are to be checked in case the preceding set of requirements is not met.

Two or more REQUIRES statements following each other immediately are considered to be in an 'AND' relation. This means that the REQUIRES check is successful only if the prerequisites, corequisites, and negative prerequisites of the whole set of REQUIRES statements are met.

**Example:**

```
REQ PRE=06645C  
OR  
REQ PRE=06645D
```

## PTF

▶▶PTF=(ptf\_number)◀◀

The PTF statement is used as a detail control statement to the LIST SERVICETAPE COVER statement to print selected cover letters.

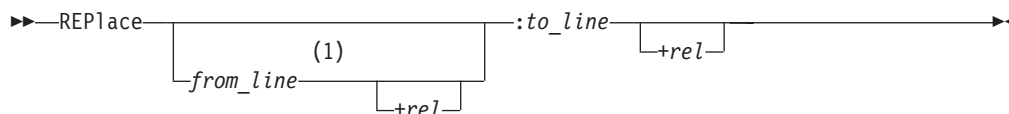
### Description of Operands

#### =ptf-number

Identifies the PTF whose cover letter is to be printed.

**Example:** PTF=(UD34634,UD38476)

## REPLACE

**Notes:**

- 1 Default: If the operand is omitted, `from_line` is assumed to be equal to the `to_line` value.

The REPLACE statement is used when applying (CORRECT) or archiving (ARCHIVE) a local or APAR fix to define where replacement of lines in a source macro must begin and end.

The replacing data must follow immediately the REPLACE statement and is to be terminated by an input line containing `/$` in columns 1 and 2 (or by a blank line when entered from SYSLOG).

**Note:** The REPLACE statement cannot be used for members of type PROC or HTML.

**Description of Operands****from-line**

Specifies, by the line-number in columns 73 through 78 in the macro, the first line to be deleted and to be replaced by the first (if any) input line. (Input refers to data that follows the REPLACE statement.)

*Default:* If `from-line` is not specified, it is assumed to be equal to `to-line`. In that case, only this line (the one designated by `to-line`) is replaced in the macro. It may, however, be replaced with more than one line of replacing data.

**+rel**

Specifies the position of the line relative to the `from-line` number.

`rel` is an integer of one or two digits.

*Restriction:* `rel` applies to E- or F-type macros only.

**to-line**

Specifies that, beginning with `from-line`, all lines in the macro are to be deleted up to and including the line indicated by `to-line`. `to-line` is the line-number contained in columns 73 through 78 of the macro to be modified.

**+rel**

Specifies the position of the line relative to the `to-line` number.

`rel` is an integer of one or two digits.

*Restriction:* `rel` applies to E- or F-type macros only.

**Example:** REPLACE 212400 : 212420



## REQUIRES

**Notes:**

- 1 At least one operand must be specified.

The REQUIRES statement is used to specify the requirements (such as prerequisite PTFs) that must be met to successfully install a shipment package or apply service in PTF or local/APAR fix format. The specified requirements are entered in the history file that accompanies the programming package.

*Restrictions:*

1. The number of requirements per PTF, local/APAR fix, component, or product specified in one or more REQUIRES statements must not exceed 88.
2. At least one of the operands PRE=, CO=, or NOT= must be present.

You may connect several requirements (with an 'AND' relation) by specifying several REQUIRES statements in succession. This means that the REQUIRES check is successful only if the requisites of *all* the REQUIRES statements are met.

You can also delimit such a set of REQUIRES statements from a preceding set by means of the OR statement. If the preceding set of requirements (at least one) fails, MSHP tests the set of requirements initiated by OR. If that test is successful, all the requirements are considered to be met.

**Description of Operands****component**

If the requirements specified in the req-lists are PTFs or local/APAR fixes, then component specifies the component to which the PTF or local/APAR fix belongs.

*Default:* If component is omitted, then the PTFs or local/APAR fix(es) specified as requirements are assumed to belong to the component to which the "requiring" PTF or local/APAR fix is applied.

*Restrictions:* Component must not be specified if the requirement in a req-list is neither a PTF nor a local/APAR fix. Component must always be specified if REQUIRES is used in conjunction with the ARCHIVE statement.

**PRE**

Indicates that the requirements specified in the req-list have to be installed *prior to* the requested service application or installation function.

A prerequisite condition is also considered as being met if a prerequisite PTF has been superseded by another, installed PTF.

**CO**

Indicates that the requirements specified in the req-list have to be applied *together with* the requested service application or installation function.

## REQUIRES

If REQUIRES is used as a detail control statement to CORRECT, CO= indicates that the requesting local/APAR fix will be applied even though the requirements specified in the req-list are not met; however, MSHP issues a warning message.

### NOT

Indicates that the requirements specified in the req-list must *not* be installed prior to the requesting service application or installation function.

### req-list

A 'req' is one of the following:

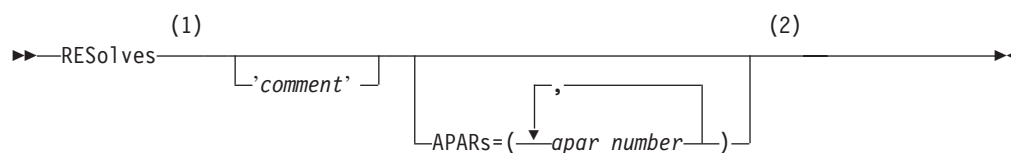
- PTF number or local/APAR fix number
- component[-level]
- product (or old feature number)

*Restriction:* In a requirements list, all items must be of the same type: PTF numbers, APAR numbers, components, and products may not be mixed.

### Examples:

```
REQUIRES 5686-CF7-06-81C PRE=DY48376
REQUIRES PRE=06645C
REQUIRES CO=DY23540
```

## RESOLVES

**Notes:**

- 1 At least one operand must be specified.
- 2 APAR=(apar\_number) must be specified in a RESOLVES statement that relates to a PTF (RESOLVES being used in conjunction with ARCHIVE PTF and APPLY component:ptf\_number).

The RESOLVES statement associates a comment with a product, a PTF, a local/APAR fix, or a generated member; it is also used to indicate which APARs are fixed by a PTF.

*Restriction:* Only one comment per associated product (or fix or member) can be recorded in the history file. If more than one RESOLVES 'comment' statement is specified, the last one will be recorded.

**Description of Operands****'comment'**

Specifies that a comment relating to a PTF, a local/APAR fix, a product, or a generated member is to be inserted in the history file.

For the function ARCHIVE product, the comment is required in the RESOLVES statement.

'comment' is a string of characters enclosed in quotes. The maximum length of the string is 35 characters if the comment is associated with a local/APAR fix; it is 57 for any other comment.

**APARs=(apar-number,...)**

Specifies the APAR number(s) corrected by a given PTF.

*Restriction:* This operand must be specified in a RESOLVES statement that relates to a PTF (RESOLVES being used in conjunction with ARCHIVE PTF and APPLY component:ptf-number).

**Example:** RESOLVES 'INIT DISK ERROR' APAR=DY45000

## RESTART

### RESTART

▶▶—REStArt *restart\_line* +rel————▶▶

The RESTART statement is used for the correction of edited macros (with the CORRECT statement). It indicates that a new sequence number series starts after the specified statement.

#### Description of Operands

##### **restart-line**

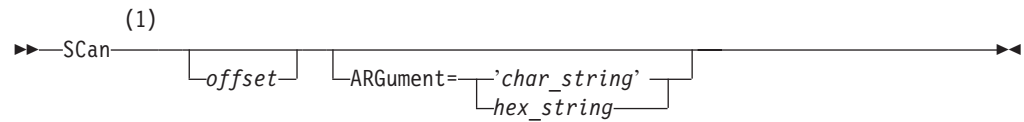
Specifies the sequence number of the statement after which the new sequence number series starts.

##### **+rel**

Specifies the position of the desired statement relative to 'restart-line'.

**Example:** RESTART 000850

## SCAN



### Notes:

- 1 At least one operand must be specified.

The SCAN statement is used when correcting a phase (after AFFECTS PHASES=...) to search for a specified string in a phase and to display 16 bytes of the phase. SCAN is not supported if the EXPAND option was specified in the AFFECTS statement.

### Description of Operands

#### offset

Specifies the displacement (relative to the beginning of the phase) where, in the phase, the search for the specified ARGUMENT string is to be started. If the ARGUMENT=string operand is omitted, MSHP displays 16 bytes of the phase, starting at 'offset'. 'offset' is a number of up to six hexadecimal digits; leading zeros may be omitted.

#### ARGUMENT='char-string' | hex-string

Specifies the string that is to be searched in the phase. It can be in one of the following formats:

- A string of 1 to 16 characters, enclosed in quotes, where each character represents one byte in the phase.
- An even string of 2 to 32 hexadecimal digits, where each pair of hexadecimal digits describes one byte in the phase.

The following table shows the results of specifying the two operands 'offset' and 'ARGUMENT' in various combinations.

	Offset	ARG	Result
First SCAN after AFFECTS PHASES	-	-	Invalid; error message.
	-	x	Scanning for specified string from offset 0.
	x	-	Display of 16 bytes from specified offset.
	x	x	Scanning for indicated string from specified offset.
Subsequent SCAN	-	-	Scanning from current offset for old argument string, which must be known from preceding scan request.
	-	x	Scanning for specified string from current offset.
	x	-	Display of 16 bytes from specified offset.
	x	x	Scanning for indicated string from specified offset.

**Example:** SCAN 0040 ARG=47500000

## SUPERSEDES

### SUPERSEDES



The SUPERSEDES statement identifies which PTFs are superseded by a given PTF when that PTF is being built.

MSHP requires the list of superseded PTFs to be complete. For example: If PTF2 supersedes PTF1, and subsequently a PTF3 is issued that supersedes PTF2, then PTF3 must be specified as also superseding PTF1.

#### Description of Operands

##### (ptf-number,...)

Specifies the PTF(s) that are superseded.

The superseded ptf-numbers are recorded in the history file entry for the superseding PTF.

The superseded PTF itself, if on the system, is marked in the history file entry as superseded.

*Restriction:* The maximum number of PTFs that can be specified as superseded in *one* SUPERSEDES statement is 100. The *total* number that can be specified is 255.

**Example:** SUPERSEDES (UD38765,UD37645)

## VERIFY

►►—VERify *verify\_line* —┐  
                                   └+rel┘

The VERIFY statement applies to source members of types E and F only. It designates where a verification is to be made for a local or APAR fix correction. The VERIFY statement must be followed by a single line of text. MSHP checks whether this text is present in the statement indicated by 'verify-line'.

### Description of Operands

#### **verify-line**

Specifies the sequence number of the source statement to be verified.

verify-line is an integer of one to six decimal digits. If fewer than six digits are coded, leading zeros are supplied.

**rel** Specifies the position of the desired statement in relation to the statement number indicated for verify-line.

rel is an integer of one or two digits.

**Example:** VERIFY 032100+25





---

## Appendix. Format of Linkage Editor Statements

---

### Format of the ESD Statement

Card Columns	Content
--------------	---------

1	X'02'. Identifies this as a statement of an object module.
---	--

2 - 4	ESD -- External Symbol Dictionary statement.
-------	--

11 - 12	Number of bytes of information contained in this statement.
---------	---

15 - 16	External symbol identification number (ESID) of the first SD, PC, CM or ER on this statement. Relates the SD, PC, CM, PR, or ER to a particular control section.
---------	--

17 - 72	Variable information.
---------	-----------------------

<b>8 positions</b>	Name
--------------------	------

<b>1 position</b>	Type code hex '00', '01', '02', '04', '05', '06', or '0A' to indicate SD, LD, ER, PC, CM, PR, or WX, respectively.
-------------------	--

<b>3 positions</b>	Assembled origin
--------------------	------------------

<b>1 position</b>	AMODE/RMODE data if SD or PC:
-------------------	-------------------------------

xxxx x...	Not used
xxxx xR..	RMODE data:
	0 = 24
	1 = ANY
xxxx xxAA	AMODE data:
	00,01 = 24
	10 = 31
	11 = ANY

Blank if LD, ER, CM, PR, or WX.

<b>3 positions</b>	Length, if an SD-type, CM-type, PR-type, or a PC-type. If an LD-type, this field contains the external symbol identification number (ESID) of the SD containing the label.
--------------------	--

73 - 80	May be used by the programmer for identification.
---------	---

---

## Format of the TXT Statement

Card Columns	Content
1	X'02'. Identifies this as a statement of an object module.
2 - 4	TXT -- Text statement.
6 - 8	Assembled origin (address of first byte to be loaded from this statement).
11 - 12	Number of bytes of text to be loaded.
15 - 16	External symbol identification number (ESID) of the control section (SD or PC) containing the text.
17 - 72	Up to 56 bytes of text -- data or instructions to be loaded.
73 - 80	May be used for program identification.

---

## Format of the RLD Statement

Card Columns	Content												
1	X'02'. Identifies this as a statement of an object module.												
2 - 4	RLD -- Relocation List Dictionary statement.												
11 - 12	Number of bytes of information contained in this statement.												
17 - 72	Variable information (multiple items). <ol style="list-style-type: none"><li>Two positions (relocation identifier) - the ESID number of the ESD item on which the relocation factor of the contents of the address constant is dependent. (Zero if cumulative pseudo-register length CxD.)</li><li>Two positions (position identifier) - the ESID number of the ESD item on which the position of the address constant is dependent.</li><li>One position - flag byte indicating type of constant, as follows:<table><thead><tr><th>Bits</th><th>Setting and Meaning</th></tr></thead><tbody><tr><td>0-1</td><td>(ignored)</td></tr><tr><td>2-3</td><td>00 - a nonbranch type load constant 01 - a branch type load constant 10 - a pseudo-register (Q-type address constant) 11 - a cumulative pseudo-register length (RLD)</td></tr><tr><td>4-5</td><td>00 - load constant length = 1 byte 01 - load constant length = 2 bytes 10 - load constant length = 3 bytes 11 - load constant length = 4 bytes</td></tr><tr><td>6</td><td>0 - relocation factor is to be added 1 - relocation factor is to be subtracted</td></tr><tr><td>7</td><td>0 - Next load constant has different R and P identifiers; therefore, both R and P must be present. 1 - Next load constant has the same R and P identifiers; therefore, they are both omitted.</td></tr></tbody></table></li><li>Three positions - assembled origin of load constant.</li></ol>	Bits	Setting and Meaning	0-1	(ignored)	2-3	00 - a nonbranch type load constant 01 - a branch type load constant 10 - a pseudo-register (Q-type address constant) 11 - a cumulative pseudo-register length (RLD)	4-5	00 - load constant length = 1 byte 01 - load constant length = 2 bytes 10 - load constant length = 3 bytes 11 - load constant length = 4 bytes	6	0 - relocation factor is to be added 1 - relocation factor is to be subtracted	7	0 - Next load constant has different R and P identifiers; therefore, both R and P must be present. 1 - Next load constant has the same R and P identifiers; therefore, they are both omitted.
Bits	Setting and Meaning												
0-1	(ignored)												
2-3	00 - a nonbranch type load constant 01 - a branch type load constant 10 - a pseudo-register (Q-type address constant) 11 - a cumulative pseudo-register length (RLD)												
4-5	00 - load constant length = 1 byte 01 - load constant length = 2 bytes 10 - load constant length = 3 bytes 11 - load constant length = 4 bytes												
6	0 - relocation factor is to be added 1 - relocation factor is to be subtracted												
7	0 - Next load constant has different R and P identifiers; therefore, both R and P must be present. 1 - Next load constant has the same R and P identifiers; therefore, they are both omitted.												
73 - 80	May be used for program identification.												

---

## Format of the END Statement

Card Columns	Content
1	X'02'. Identifies this as a statement of an object module.
2 - 4	END
6 - 8	Assembled origin of the label supplied to the assembler in the END statement (optional).
15 - 16	ESID number of the control section to which this END statement refers (only if 6-8 present).
17 - 22	Symbolic label supplied to the assembler if this label was not defined within the assembly.
29 - 32	Control section length (if not specified in last SD or PC).
73 - 80	Not used.

---

## Format of the REP (User Replace) Statement

Card Columns	Content
1	X'02'. Identifies this as a statement of an object module.
2 - 4	REP -- Replace text statement.
5 - 6	Blank.
7 - 12	Assembled address of the first byte to be replaced (hexadecimal). Must be right justified with leading zeros if needed to fill the field and must be equal to or greater than the starting address of the control section (columns 14-16). Note that there is no check to determine if the assembled address is actually within this control section.
13	Blank.
14 - 16	External symbol identification number (ESID) of the control section (SD) containing the text (hexadecimal). Must be right justified with leading zeros if needed to fill the field.
17 - 70	From 1 to 11 4-digit hexadecimal fields separated by commas, each replacing two bytes. A blank indicates the end of information in this statement.
71 - 72	Blank.
73 - 80	May be used for program identification.

---

## External Symbol Dictionary

The external symbol dictionary (ESD) contains control section definitions and inter-module references. Seven types of entries are defined in the control dictionary:

<b>ESD Type</b>	<b>Definition</b>
<b>SD</b>	Section definition: provides control section name, assembled origin and length. Generated by a named START or a named CSECT in a source module.
<b>WX</b>	Generated by weak external reference (WXTRN), which has a function similar to EXTRN, except that WXTRN suppresses AUTOLINK. The linkage editor treats WX as an ER, NOAUTO.
<b>PC</b>	Private code: provides assembled origin and length for an unnamed control section.
<b>LD/LR</b>	Label definition: specifies the assembled address and the associated SD of a label that may be referred to by another module. The LD entry is termed LR (Label Reference) when the entry is matched to an ER entry.
<b>ER</b>	External reference: specifies the location of a reference made to another module. ER is generated by EXTRN or a V-type address constant in a source module.
<b>CM</b>	Common: indicates the amount of storage to be reserved for common use by different phases. CM is generated by COM in a source module.
<b>PR</b>	Pseudo-register: provides name, alignment, and length of storage that can be allocated during execution. (Assembler term: external dummy section.)

---

## Glossary

This glossary includes terms and definitions related primarily to IBM z/VSE. For z/VSE component programs whose terms are not included in this glossary, such as VTAM or CICS/VSE, refer to the respective General Information manuals.

If you do not find the term you are looking for, refer to the index of this book or to the *IBM Dictionary of Computing, SC20-1699*

The glossary includes definitions with:

- Symbol \* where there is a one-to-one copy from the IBM Dictionary of Computing.
- Symbol (A) from the *American National Dictionary for Information Processing Systems*, copyright 1982 by the Computer and Business Equipment Manufacturers Association (CBEMA). Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- Symbols (I) or (T) from the *ISO Vocabulary - Information Processing* and the *ISO Vocabulary - Office Machines*, developed by the International Organization for Standardization, Technical Committee 97, Subcommittee 1. Definitions of published segments of the vocabularies are identified by the symbol (I) after the definition; definitions from draft international standards, draft proposals, and working papers in development by the ISO/TC97/SC1 vocabulary subcommittee are identified by the symbol (T) after the definition, indicating final agreement has not yet been reached among participating members.

The following cross-references are used:

- Contrast with. This refers to a term that has an opposed or substantively different meaning.
- Synonym for. This indicates that the term has the same meaning as a preferred term, which is defined in its proper place in the dictionary.
- Synonymous with. This is a backward reference from a defined term to all other terms that have the same meaning.
- See. This refers the reader to multiple-word terms that have the same last word.

- See also. This refers the reader to related terms that have a related, but not synonymous, meaning.

When an entry is an abbreviation, the explanation consists of the spelled-out meaning of the abbreviation, for example:

*AFP*. Advanced Function Printing.

The spelled-out form is provided as a separate entry in the glossary. In that entry, the abbreviation is shown in parentheses after the spelled-out form. The definition that appears with the spelled-out entry provides the full meaning of both the abbreviation and the spelled-out form:

*Advanced Function Printing (AFP)*. A group of...

**access control.** A function of VSE that ensures that the system and the data and programs stored in it can be accessed only by authorized users in authorized ways.

\* **Access Control - Logging and Reporting.** An IBM licensed program used to log access to protected data and to print selected formatted reports on such access.

**access list.** A table in which each entry specifies an address space or data space that a program can reference.

**access method.** A program, that is, a set of commands (macros), to define files or addresses and to move data to and from them; for example VSE/VSAM or VSE/VTAM.

**access register (AR).** A hardware register that a program can use to identify an address space or a data space. Each processor has 16 ARs, numbered 0 through 15, which are paired one-to-one with the 16 general-purpose registers (GPRs).

\* **account file.** A direct access file maintained by VSE/POWER to hold the accounting information it generates and the programs that it controls.

**ACF/VTAM.** See VTAM.

**addressing mode (AMODE).** A program attribute that refers to the address length that a program is prepared to handle on entry. Addresses may be either 24 bits or 31 bits in length. In 24-bit addressing mode, the processor treats all virtual addresses as 24-bit values; in 31-bit addressing mode, the processor treats all virtual addresses as 31-bit values. Programs with an addressing mode of ANY can receive control in either 24-bit or 31-bit addressing mode.

**address space.** A range of up to two gigabytes of contiguous virtual storage addresses that the system creates for a user. Unlike a data space, an address space contains user data **and** programs, as well as system data and programs, some of which are common to all

address spaces. Instructions execute in an address space (not a data space). Contrast with data space.

**address space control (ASC) mode.** The mode (determined by the PSW) that tells the system where to find referenced data. It determines how the processor resolves address references for the executing programs. VSE/ESA supports two types of ASC modes:

1. In **primary** ASC mode, the data that a program can access resides in the program's own (primary) address space. In this mode, the system uses the contents of general-purpose registers to resolve an address in the address space; it does not use the contents of the access registers (ARs).
2. In **access register (AR)** ASC mode, the data that a program can access may reside in an address space other than the primary or in a data space. In this mode, the system uses both a general-purpose register (GPR) and the corresponding access register together to resolve an address in another address space or in a data space. Specifically, the AR contains a value, called an ALET, that identifies the address space or data space that contains the data, and the GPR contains a base address that points to the data within the address space or data space.

\* **Advanced Communications Function (ACF).** A group of IBM licensed programs, principally VTAM programs, that use the concepts of Systems Network Architecture (SNA), including distribution of function and resource sharing.

**Advanced Function Printing™ (AFP).** A group of IBM licensed programs that support APA printers.

**AFP.** Advanced Function Printing.

**ALET (access list entry token).** A token that points to an entry in an access list. When a program is in AR mode and the ALET is in an access register (with the corresponding general-purpose register being used as base register), the ALET identifies the address space or data space that the system is to reference (while the GPR indicates the offset within the space).

\* **all points addressable (APA).** In computer graphics, pertaining to the ability to address and display or not display each picture element (pel) on a display surface.

**alternate block.** On an FBA disk, a block designated to contain data in place of a defective block.

\* **alternate index.** In systems with VSE/VSAM, a collection of index entries related to a given base cluster and organized by an alternate key, that is, a key other than the prime key of the base cluster data records; it gives an alternate directory for finding records in the data component of a base cluster. See also *path*.

\* **alternate tape.** A tape drive to which the operating system switches automatically for tape read or write operations if the end of the volume has been reached on the originally used tape drive.

\* **alternate track.** On a direct access device, a track designated to contain data in place of a defective primary track.

\* **American National Standard Code for Information Interchange (ASCII).** The standard code, using a coded character set consisting of 7-bit coded characters

(8 bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

**AMODE.** Addressing mode.

**APA.** All points addressable.

**APAR.** Authorized program analysis report.

\* **appendage routine.** Code physically located in a program or subsystem, but logically an extension of a VSE supervisor routine.

\* **application profile.** A control block in which the system stores the characteristics of one or more application programs.

\* **application program.** A program written for or by a user that applies directly to the user's work, such as a program that does inventory control or payroll. See also *batch program* and *online application program*.

**AR (access register) mode.** If a program runs in AR mode, the system uses the access register/general-purpose register pair to resolve an address in an address space or data space. Contrast with *primary mode*. See also *address space control (ASC) mode*.

**ASC mode.** Address space control mode.

**ASI (automated system initialization) procedure.** A set of control statements which specifies values for an automatic system initialization.

\* **assemble.** To translate an assembly language program into an object program. (T)

\* **assembler.** A computer program that converts assembly language instructions into object code.

**assembler language.** A programming language whose instructions are usually in one-to-one correspondence with machine instructions and allows to write macros.

**attention routine.** A routine of the system that receives control when the operator presses the Attention key. The routine sets up the console for the input of a command, reads the command, and initiates the system service requested by the command.

\* **authorized program analysis report (APAR).** A request for a correction of a problem caused by a defect in a current unaltered release of a program.

**autolink.** An automatic library look-up function of the linkage editor. The function a) resolves any external reference that is included in the currently processed module and b) searches the active phase-search chain for an object module of the same name as the encountered external reference.

\* **automated system initialization (ASI).** A function that allows control information for system startup to be cataloged for automatic retrieval during system startup.

\* **auxiliary storage.** All addressable storage, other than main storage, that can be accessed by means of an input/output channel; for example storage on magnetic tape or direct access devices. Synonymous with *external storage*.

**background partition.** An area of virtual storage in which programs are executed under control of the system. By default, the partition has a processing priority lower than any of the existing foreground partitions.

**backup copy.** A copy, usually of a file or a library member, that is kept in case the original file or library member is unintentionally changed or destroyed.

\* **base cluster.** In systems with VSAM, a key-sequenced or entry-sequenced file over which one or more alternate indexes are built.

**basic telecommunications access method (BTAM).** An access method that permits read and write communication with remote devices. Its current version is called BTAM-ES.

**batch processing.** 1. Serial processing of computer programs. 2. Pertaining to the technique of executing a set of computer programs such that each is completed before the next program of the set is started. (A)

**batch program.** A program that is processed in series with other programs and therefore normally processes data without user interaction.

**binary synchronous communication (BSC).** Method of telecommunication using binary synchronous line discipline. Contrast with *SDLC*.

**bits per second (bps).** In serial transmission, the instantaneous bit speed with which a device or channel transmits a character.

**block.** Usually, a block consists of several records of a file that are transmitted as a unit. But if records are very large, a block can also be part of a record only. See also *control block*.

**blocking.** The process of combining (or cutting) records into blocks.

**bps.** Bits per second.

\* **bringup.** The process of starting a computer system or a subsystem that is to operate under control of the system.

\* **BTAM-ES (Basic Telecommunication Access Method Extended Storage).** An IBM supplied telecommunication access method that permits read and write communication with remote devices.

**B-transient.** A phase with a name beginning with \$\$B and running in the Logical Transient Area (LTA). Such a phase is activated by special supervisor calls.

**cache storage.** A random access electronic storage in selected storage controls used to retain frequently used data for faster access by the channel. For example, the IBM 3990 Model 3 contains cache.

\* **catalog.** 1. A directory of files and libraries, with reference to their locations. A catalog may contain other information such as the types of devices in which the files are stored, passwords, blocking factors. (I) (A) 2. To store a library member such as a phase, module, or book in a sublibrary.

\* **cataloged procedure.** A set of control statements placed in a library and retrievable by name.

**CCB.** Command control block.

**CCW.** Channel command word.

\* **central location.** The place at which a computer system's control device, normally the system console in the computer room, is installed.

**Central Processing Complex.** A segment of the physical resources of a system configuration. In this segment, a supervisor operates as it would operate on a dedicated system.

**central processing unit (CPU).** The hardware component that interprets and executes instructions. Synonym for *processor*.

\* **chained sublibraries.** A facility that allows sublibraries to be chained by specifying the sequence in which these sublibraries are to be searched for a certain library member of a certain type.

\* **chaining.** A logical connection of sublibraries to be searched by the system for members of the same type; for example, phase or object modules.

\* **channel adapter.** A communication controller hardware unit used to attach the controller to a System/370™ data channel.

\* **channel-attached.** Pertaining to attachment of devices directly by data channels (I/O channels) to a computer. Contrast with *link-attached*. Synonymous with *locally attached*.

\* **channel command word (CCW).** A doubleword at the location in main storage specified by the channel address word. One or more CCWs make up the channel program that directs data channel operations.

\* **channel program.** One or more channel command words that control a sequence of data channel operations. Execution of this sequence is initiated by a single start I/O (SIO) instruction.

\* **channel scheduler.** The part of the supervisor that controls all input/output operations.

**channel subsystem.** A feature of 370-XA and Enterprise Systems Architecture that provides extensive additional channel (I/O) capabilities over the System/370.

**character printer.** A device that prints a single character at a time. (T) (A) Contrast with *line printer*, *page printer*.

\* **checkpoint.** 1. A point at which information about the status of a job and the system can be recorded so that the job step can be restarted later. 2. To record such information.

**CI.** Control interval.

**CICS.** Customer Information Control System.

**CKD device.** Count-key-data device.

**CMS.** Conversational monitor system.

**COBOL.** Common business-oriented language.

**command control block (CCB).** The name of a system control block to hold information about a specific instance of a command.

**common business-oriented language (COBOL).** A high-level programming language based on English used primarily for business application programs.

\* **common library.** An interactively accessible library that can be accessed by any user of the system or subsystem that owns the library.

\* **communication adapter.** A circuit card with associated software that enables a processor, controller, or other device to be connected to a network.

\* **communication controller.** 1. A device that directs the transmission of data over the data links of a network; its operation may be controlled by a program executed in a processor to which the controller is connected or it may be controlled by a program executed within the device. (T) 2. A type of

communication control unit whose operations are controlled by one or more programs stored and executed in the unit. It manages the details of line control and the routing of data through a network.

**communication line.** See *telecommunication line*.

\* **communication region.** An area of the supervisor that is set aside for transfer of information within and between programs.

\* **compaction.** In SNA, the transformation of data by packing two characters in a byte so as to take advantage of the fact that only a subset of the allowable 256 characters is used; the most frequently sent characters are compacted.

\* **compile.** To translate a source program into an executable program (an object program). See also *assembler*.

**compiler.** A program used to compile.

**component.** 1. Hardware or software that is part of a computer system. 2. A functional part of a product, identified by a component identifier. 3. In VSE/ESA, a component program such as VSE/POWER or VTAM. 4. In VSE/VSAM, a named, cataloged group of stored records, such as the data component or index component of a key-sequenced file or alternate index.

**computer system.** A functional unit of one or more computers and their associated software.

**conditional job control.** The capability of the job control program to process or to skip one or more statements based on a condition that is tested by the program.

\* **configuration.** The devices and programs that make up a system, subsystem, or network.

**connect.** To authorize library access on the lowest level. A modifier such as "read" or "write" is required for the specified use of a sublibrary.

**control block.** An area within a program or a routine defined for the purpose of storing and maintaining control information.

\* **control interval (CI).** A fixed-length area of disk storage where VSE/VSAM stores records and distributes free space. It is the unit of information that VSE/VSAM transfers to or from disk storage. For FBA, it must be an integral multiple, to be defined at cluster definition, of the block size.

**control program.** A program to schedule and supervise the running of programs in a system.

**control unit.** See *communication controller*. Synonymous with *controller*.

\* **conversational monitor system (CMS).** A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities, and operates only under control of the VM/370 VM control program.

\* **corrective service.** The installation of a PTF or an APAR fix that corrects a specific problem.

**count-key-data (CKD) device.** A disk device that stores data in the record format: count field, key field, data field. The count field contains, among others, the address of the record in the format: cylinder, head (track), record number and the length of the data field. The key field, if present, contains the record's key or

search argument. CKD disk space is allocated by tracks and cylinders. Contrast with *FBA disk device*. See also *extended count-key-data device*.

\* **cross-partition communication control.** A facility that enables VSE subsystems and user programs to communicate with each other; for example, with VSE/POWER.

**Customer Information Control System (CICS).** An IBM licensed program that controls online communication between terminal users and a database. Transactions entered at remote terminals are processed concurrently by user-written application programs. The product includes facilities for building, using, and servicing databases.

Its current version is called CICS/VSE.

**DASD.** Direct access storage device.

**DASD sharing.** An option that lets independent computer systems use common data on shared disk devices.

**database.** A set of data available online that is organized by a common system and used for a common purpose.

\* **data entry panel.** A panel in which the user communicates with the system by filling in one or more fields. See also *panel* and *selection panel*.

**data file.** See *file*.

**data import.** The process of reformatting data that was used under one operating system (for example, IBM System/3) such that it can subsequently be used under a different operating system (for example, the VSE system).

\* **Data Interfile Transfer, Testing and Operations (DITTO) utility.** An IBM licensed program that provides file-to-file services for card I/O, tape, and disk devices.

**Data Language/I (DL/I).** A database access language used with CICS/VSE.

**data link.** In SNA, the combination of the link connection and the link stations joining network nodes, for example, a System/370 channel and its associated protocols. A link is both logical and physical.

In SNA, synonym for *link*.

\* **data management.** A major function of the operating system. It involves organizing, storing, locating, and retrieving data.

**data processing system.** Synonym for *computer system*.

**data security.** See *access control*.

**data set.** See *file*.

**data space.** A range of up to two gigabytes of contiguous virtual storage addresses that a program can directly manipulate through ESA/370 instructions. Unlike an address space, a data space can hold only user data; it does not contain shared areas, system data or programs. Instructions do not execute in a data space, although a program can reside in a data space as non-executable code. Contrast with address space.

\* **deblocking.** The process of making each logical record of a block available for processing. Contrast with *blocking*.



**default value.** A value assumed by the program when no value has been specified by the user.

**definition statement.** In VTAM, the means of describing an element of the network.

\* **device address.** 1. The identification of an input/output device by its device number. 2. In data communication, the identification of any device to which data can be sent or from which data can be received.

\* **device class.** The generic name for a group of device types; for example, all display stations belong to the same device class. Contrast with *device type*.

\* **Device Support Facilities.** An IBM supplied system control program for performing operations on disk volumes so that they can be accessed by IBM and user programs. Examples of these operations are initializing a disk volume and assigning an alternate track.

\* **device type code.** The four- or five-digit code to be used for defining an I/O device to a computer system.

\* **dialog.** 1. In an interactive system, a series of related inquiries and responses similar to a conversation between two people. 2. For VSE/ESA, a set of panels that can be used to complete a specific task; for example, defining a file.

**direct access.** Accessing data on a storage device using their address and not their sequence. This is the typical access on disk devices as opposed to magnetic tapes. Contrast with *sequential access*.

\* **Direct access storage device (DASD).** A device in which access time is effectively independent of the location of the data.

**directory.** 1. A table of identifiers and references to the corresponding items of data. (I) (A) 2. In VSE, specifically, the index for the program libraries. See also *library directory* and *sublibrary directory*.

**disk operating system residence volume (DOSRES).** The disk volume on which the system sublibrary IJSYSRS.SYSLIB is located including the programs and procedures required for system startup.

**disk sharing.** An option that lets independent computer systems use common data on shared disk devices.

**display station.** A display screen with attached keyboard for communication with the system or a network. See also *terminal*.

**disposition.** A means of indicating to VSE/POWER how job input and output is to be handled. A job may, for example, be deleted or kept after processing.

\* **distribution tape.** A magnetic tape that contains, for example, a preconfigured operating system like VSE/ESA. This tape is shipped to the customer for program installation.

\* **DITTO utility.** Data Interfile Transfer, Testing and Operations utility.

**DL/I.** Data Language/I.

**domain.** The network resources under the control of a particular SSCP.

**DOSRES.** Disk operating system residence volume.

**DU-AL (dispatchable unit - access list).** The access list that is associated with a VSE/ESA main task or

subtask. A program uses the DU-AL associated with its task and the PASN-AL associated with its partition. See also PASN-AL.

**dummy device.** A device address with no real I/O device behind it. Input and output for that device address are spooled on disk.

\* **dump.** 1. Data that has been dumped. (T) 2. To record, at a particular instant, the contents of all or part of one storage device in another storage device.

Dumping is usually for the purpose of debugging. (T)

\* **duplex.** Pertaining to communication in which data can be sent and received at the same time.

**dynamic class table.** Defines the characteristics of dynamic partitions.

**dynamic partition.** A partition created and activated on an 'as needed' basis that does not use fixed static allocations. After processing, the occupied space is released. Contrast with *static partition*.

\* **dynamic partition balancing.** A VSE facility that allows the user to specify that two or more or all partitions of the system should receive about the same amount of time on the processing unit.

\* **dynamic space reclamation.** A librarian function that provides for space freed by the deletion of a library member to become reusable automatically.

**EBCDIC.** Extended binary-coded decimal interchange code.

**ECKD device.** Extended count-key-data device.

**emulation.** The use of programming techniques and special machine features that permit a computer system to execute programs written for another system or for the use of I/O devices different from those that are available.

**end user.** 1. A person who makes use of an application program. 2. In SNA, the ultimate source or destination of user data flowing through an SNA network. May be an application program or a terminal operator.

**Enterprise Systems Architecture (ESA).** See *ESA/370* and *ESA/390*.

\* **entry-sequenced file.** A VSE/VSAM file whose records are loaded without respect to their contents and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the file.

**Environmental Record Editing and Printing (EREP)**

**program.** The program that makes the data contained in the system recorder file available for further analysis.

**EREP program.** Environmental Record Editing and Printing program.

\* **error recovery procedures (ERP).** Procedures designed to help isolate and, where possible, to recover from errors in equipment. :gt.ESA mode :gd.An operation mode of the supervisor (generated with MODE=ESA) of a VSE system. Such a supervisor will run on a 370=XA or Enterprise Systems Architecture processor and provides support for multiple virtual address spaces, the channel subsystem, and more than 16MB of real storage.

**ESA/370.** IBM Enterprise Systems Architecture/370™. The extension to the IBM System/370 architecture which includes the advanced addressability feature that provides access registers.

**ESA/390.** IBM Enterprise Systems Architecture/390®. The latest extension to the IBM System/370 architecture which includes the advanced addressability feature and advanced channel architecture.

\* **escape.** To return to the original level of a user interface.

**extended addressability.** 1. See *31-bit addressing*. 2. The ability of a program to use virtual storage that is outside the address space in which the program is running. Generally, instructions and data reside in a single address space - the primary address space. However, a program can have data in address spaces other than the primary or in data spaces. (The instructions remain in the primary address space, whilst the data can reside in another address space or in a data space.) To access data in other address spaces, a program must use access registers (ARs) and execute in access register mode (AR mode).

**extended binary-coded decimal interchange code (EBCDIC).** A coded character set consisting of 8-bit coded characters.

**extended count-key-data (ECKD) device.** A disk storage device that has a data transfer rate faster than some processors can utilize. A specialized channel program is needed to convert ordinary CKD channel programs for use with an ECKD device.

**extent.** Continuous space on a disk or diskette occupied by or reserved for a particular file or VSAM data space.

**external storage.** Storage that is not part of the processor.

**FASTCOPY.** A stand-alone utility for fast copy data operations from disk to disk and dump/restore operations via an intermediate dump file on magnetic tape or disk. There is also an online version: the Fast Copy Data Set program. Acronym = VSE/FCOPY

**Fast Copy Data Set program (VSE/FCOPY).** See *VSE/FCOPY*.

**FBA disk device.** Fixed-block architecture disk device.

**FCB.** Forms control buffer.

\* **file.** A named set of records stored or processed as a unit. (T) Synonymous with *data set*.

**fixed-block architecture (FBA) disk device.** A disk device that stores data in blocks of fixed size. These blocks are addressed by block number relative to the beginning of the file. Contrast with *CKD device*.

\* **foreground partition.** A space in virtual storage in which programs are executed under control of the system. By default, a foreground partition has a higher processing priority than the background partition.

\* **forms control buffer (FCB).** In the 3800 Printing Subsystem, a buffer for controlling the vertical format of printed output.

\* **fragmentation (of storage).** In virtual system, inability to assign real storage locations to virtual storage addresses because the available spaces are smaller than the page size.

\* **generate.** To produce a computer program by selecting subsets of skeletal code under the control of parameters. (A)

**generation.** See *macro generation*.

**generation feature.** An IBM licensed program order option used to tailor the object code of a program to user requirements.

\* **GETVIS space.** Storage space within a partition or the shared virtual area, available for dynamic allocation to programs.

**gigabyte (GB).** 1 024MB or 1 073 741 824 bytes.

**guest system.** A data processing system that runs under control of another (host) system.

\* **half-duplex.** In data communication, pertaining to transmission of data in only one direction at a time. Contrast with *duplex*.

**hard wait.** The condition of a processor when all operations are suspended. System recovery from a hard wait is impossible without performing a new system startup.

\* **hardware.** All or part of the physical components of an information processing system, such as computers or peripheral devices. (T) (A) Contrast with *software*.

\* **help panel.** A display of information provided by the system in response to a user's help request.

\* **host system.** The controlling or highest level system in a data communication configuration.

**ICA.** Integrated communication adapter.

**ICCF.** See *VSE/ICCF*.

\* **include function.** A function that retrieves a library member for inclusion in program input.

\* **initial program load (IPL).** The process of loading system programs and preparing the system to run jobs.

\* **input/output control system (IOCS).** A group of routines provided by IBM for handling transfer of data between main storage and auxiliary storage devices.

**integrated communication adapter (ICA).** The part of a processor where multiple lines can be connected.

**integrated console.** In VSE/ESA, the service processor console available on ES/9000® processors that operates as the VSE/ESA system console. The integrated console is typically used during IPL and for recovery purposes when no other console is available.

**intelligent workstation.** See *programmable workstation*.

**interactive.** A characteristic of a program or system that alternately accepts input and then responds. An interactive system is conversational, that is, a continuous dialog exists between user and system. Contrast with *batch*.

**Interactive Computing and Control Facility (ICCF).**

An IBM licensed program that serves as interface, on a time-slice basis, to authorized users of terminals linked to the system's processor.

**interactive interface.** A system facility which controls how different users see and work with the system by means of user profiles. When signing on, the interactive interface makes available those parts of the system authorized by the profile. The interactive interface has sets of selection- and data-entry panels through which users communicate with the system.

**interactive partition.** An area of virtual storage for the purpose of processing a job that was submitted interactively via VSE/ICCF.

**interface.** A shared boundary between two hardware or software units, defined by common functional or physical characteristics. It might be a hardware component or a portion of storage or registers accessed by several computer programs.

\* **intermediate storage.** Any storage device used to hold data temporarily before it is processed. See also *buffer storage*.

**IOCS.** Input/output control system.

**IPL.** Initial program load.

\* **irrecoverable error.** An error for which recovery is impossible without use of recovery techniques external to the computer program or run. (T)

**JCL.** Job control language.

**JECL.** Job entry control language.

**job accounting.** A system function that lists how much every job step uses of the different system resources.

\* **job accounting interface.** A function that accumulates accounting information for each job step that can be used for charging usage of the system, planning new applications, and supervising system operation more efficiently.

\* **job accounting table.** An area in the supervisor where accounting information is accumulated for the user.

\* **job catalog.** A catalog made available for a job by means of the filename IJSYSUC in the respective DLBL job control statement.

**job control language (JCL).** A language that serves to prepare a job or each job step of a job to be run. Some of its functions are: to identify the job, to determine the I/O devices to be used, set switches for program use, log (or print) its own statements, and fetch the first phase of each job step.

**job control statement.** A particular statement of JCL.

**job entry control language (JECL).** A control language that allows the programmer to specify how VSE/POWER should handle a job.

**job step.** One of a group of related programs complete with the JCL statements necessary for a particular run. Every job step is identified in the job stream by an EXEC statement under one JOB statement for the whole job.

**job stream.** The sequence of jobs as submitted to an operating system.

**key sequence.** The collating sequence either of records themselves or of their keys in the index or both. The key sequence is alphanumeric.

**key-sequenced file.** A VSE/VSAM file whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in the file in key sequence.

\* **kilobyte (KB).** 1024 bytes.

**label information area.** An area on a disk to store label information read from job control statements or commands. Synonymous with *label area*.

\* **language translator.** A general term for any assembler, compiler, or other routine that accepts statements in one language and produces equivalent statements in another language.

\* **librarian.** The set of programs that maintains, services, and organizes the system and private libraries.

**library.** See *VSE library* and *VSE/ICCF library*.

\* **library block.** A block of data stored in a sublibrary.

\* **library directory.** The index that enables the system to locate a certain sublibrary of the accessed library.

\* **library member.** The smallest unit of data that can be stored in and retrieved from a sublibrary.

\* **licensed program.** A separately priced program and its associated materials that bear an IBM copyright and are offered to customers under the terms and conditions of the IBM Customer Agreement (ICA).

\* **line printer.** A device that prints a line of characters as a unit. (I) (A) Contrast with *character printer* or *page printer*.

**link.** 1. To connect items of data or portions of programs; for example, linking of object programs by the linkage editor or linking of data items by pointers. 2. In SNA, the combination of the link connection and the link stations joining network nodes, for example, a System/370 channel and its associated protocols. A link is both logical and physical. Synonymous with *data link*.

\* **linkage editor.** A program used to create a phase (executable code) from one or more independently translated object modules, from one or more existing phases, or from both. In creating the phase, the linkage editor resolves cross references among the modules and phases available as input. The program can catalog the newly built phases.

\* **link-attached.** Pertaining to devices connected to a control unit by a data link. Synonymous with *remote*. Contrast with *channel-attached*.

**link connection.** Physical medium of transmission, for example a telephone wire or a microwave beam. In SNA, the physical communication equipment between link stations, for example a line and a DCE. Synonymous with *data circuit*.

**link-edit.** To create a loadable computer program by having the linkage editor process compiled (assembled) source programs.

**loader.** A routine, commonly a computer program, that reads data or a program into processor storage. See also *relocating loader*.

\* **local address.** In SNA, an address used in a peripheral node in place of a network address and transformed to or from a network address by the boundary function in a subarea node.

**Local shared resources (LSR).** A VSE/VSAM option activated by three extra macros to share control blocks among files.

\* **lock file.** In a shared disk environment under VSE, a system file on disk used by the sharing systems to control their access to shared data.

\* **logging.** The recording of data about specific events.

**logical record.** A user record, normally pertaining to a single subject and processed by data management as a unit. Contrast with *physical record* which may be larger or smaller.

**logical unit (LU).** A name used in programming to represent an I/O device address.

\* **logical unit name.** In programming, a name used to represent the address of an input/output unit.

**LSR.** local shared resources.

\* **main task.** The main program within a partition in a multiprogramming environment.

**master console.** In VSE/ESA, one or more consoles that receive all system messages, except for those that are directed to one particular console. Contrast this with the *user console* which receives only those messages that are specifically directed to it, for example messages issued from a job that was submitted with the request to echo its messages to that console. The operator of a master console can reply to all outstanding messages and enter all system commands.

\* **maintain system history program (MSHP).** A program used for automating and controlling various installation, tailoring, and service activities for a VSE system.

\* **megabyte (MB).** 1 024 KB or 1 048 576 bytes.

\* **member.** The smallest unit of data that can be stored in and retrieved from a sublibrary. See also *library member*.

**message.** 1. In VSE, a communication sent from a program to the operator or user. It can appear on a console, a display terminal or on a printout. 2. In telecommunication, a logical set of data being transmitted from one node to another.

\* **microcode.** 1. A code, representing the instructions of an instruction set, that is implemented in a part of storage that is not program-addressable. 2. To design write, and test one or more microinstructions.

\* **migrate.** To move to a changed operating environment, usually to a new release or version of a system.

\* **module.** A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from an assembler, compiler, linkage editor, or executive routine. (A)

**MSHP.** Maintain system history program.

\* **multiprogramming.** 1. A mode of operation that provides for interleaved execution of two or more computer programs by a single processor. (I) (A) 2. Pertaining to concurrent execution of two or more computer programs by a computer. (A)

**multitasking.** Concurrent running of one main task and one or several subtasks in the same partition.

**NetView®.** An IBM licensed program used to monitor a network, manage it, and diagnose its problems.

**network.** 1. An arrangement of nodes (data stations) and connecting branches. 2. The assembly of equipment through which connections are made between data stations.

**networking.** Making use of the services of a network program.

**nonprogrammable workstation (NPWS).** A workstation that does not have processing capability and that does not allow the user to change its functions. Contrast with *programmable workstation*.

\* **object code.** Output from a compiler or assembler which is itself executable machine code or is suitable for processing to produce executable machine code. (A)

**object module (program).** A program unit that is the output of an assembler or compiler and is input to a linkage editor.

**online processing.** Processing by which the input data enters the computer directly from a display station and the output data is transmitted directly to the display station.

\* **operating system.** Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible. (T)

\* **operator command.** A statement to a control program, issued via a console or terminal. It causes the control program to provide requested information, alter normal operations, initiate new operations, or end existing operations.

**optical reader/sorter.** A device that reads hand written or machine printed symbols on a voucher and, after having read the voucher, can sort it into one of the available stacker-select pockets.

**optional licensed program.** An IBM licensed program that a user can install on VSE by way of available installation-assist support.

**page data set (PDS).** One or more extents of disk storage in which pages are stored when they are not needed in processor storage.

**page fault.** A program interruption that occurs when a program page marked "not in processor storage" is referred to by an active page.

\* **page fixing.** Marking a page so that it is held in processor storage until explicitly released. Until then, it cannot be paged out.

**page frame.** An area of processor storage that can contain a page.

**page-in.** The process of transferring a page from the PDS to processor storage.

\* **page I/O.** Page-in and page-out operations.

**page-out.** The process of transferring a page from processor storage to the PDS.

\* **page pool.** The set of page frames available for paging virtual-mode programs.

**page printer.** A device that prints one page as a unit; for example, a laser printer. Contrast with *character printer, line printer*.

**panel.** The complete set of information shown in a single display on a terminal screen. Scrolling back and forth through panels is like turning manual pages. See also *selection panel* and *data entry panel*.

**partition.** A division of the virtual address area available for running programs. See also *dynamic partition, static partition*.

\* **partition balancing, dynamic.** A VSE facility that allows the user to specify that two or more or all partitions of the system should receive about the same amount of time on the processor.

**PASN-AL (primary address space number - access list).** The access list that is associated with a partition. A program uses the PASN-AL associated with its partition and the DU-AL associated with its task (work unit). See also *DU-AL*.

Each partition has its own unique PASN-AL. All programs running in this partition can access data spaces through the PASN-AL. Thus a program can create a data space, add an entry for it in the PASN-AL, and obtain the ALET that indexes the entry. By passing the ALET to other programs in the partition, the program can share the data space with other programs running in the same partition.

\* **path.** In VSAM, a named logical entity providing access to the records of a base cluster either directly or through an alternate index.

**PDS.** Page data set.

\* **physical record.** The amount of data transferred to or from auxiliary storage. Synonymous with *block*.

\* **pregenerated operating system.** An operating system such as VSE/ESA that is shipped by IBM mainly in object code. IBM defines such key characteristics as the size of the main control program, the organization and size of libraries, and required system areas on disk. The customer does not have to generate an operating system.

\* **preventive service.** The installation of one or more PTFs on a VSE system to avoid the occurrence of anticipated problems.

**primary address space.** In VSE/ESA, the address space where a partition is currently executed. A program in primary mode fetches data from the primary address space.

\* **primary library.** A VSE library owned and directly accessible by a certain terminal user.

**primary mode.** If a program runs in primary mode, the system resolves all addresses within the current (primary) address space. Contrast with *AR (access register) mode*. See also *address space control (ASC) mode*.

**Print Services Facility™ (PSF)/VSE.** An access method that provides support for the advanced function printers.

**priority.** A rank assigned to a partition or a task that determines its precedence in receiving system resources.

**private area.** The part of an address space that is available for the allocation of private partitions. Its maximum size can be defined during IPL. Contrast with *shared area*.

\* **private library.** A user-owned library that is separate and distinct from the system library.

\* **private partition.** Any of the system's partitions that are not defined as shared. See also *shared partition*.

**procedure.** See *cataloged procedure*.

\* **processing.** The performance of logical operations and calculations on data, including the temporary retention of data in processor storage while this data is being operated upon.

\* **processor.** In a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and an arithmetic and logic unit. (T)

**processor storage.** The storage contained in one or more processors and available for running machine instructions. Synonymous with *real storage*.

\* **production library.** 1. In a pre-generated operating system (or product), the program library that contains the object code for this system (or product). 2. A library that contains data needed for normal processing.

Contrast with *test library*.

**profile.** A description of the characteristics of a user or a computer resource.

\* **programmable workstation.** A workstation that has some degree of processing capability and that allows the user to change its functions. Contrast with *nonprogrammable workstation*.

\* **programmer logical unit.** A logical unit available primarily for user-written programs. See also *logical unit name*.

**program product.** See *licensed program*.

**program service.** The customer- or program-related IBM service of correcting design or implementation errors via APARs and PTFs.

**program temporary fix (PTF).** A solution or by-pass of one or more problems documented in APARs. PTFs are distributed to IBM customers for preventive service to a current release of a program.

**prompt.** To issue messages to a terminal or console user, requesting information necessary to continue processing.

**PSF/VSE.** Print Services Facility/VSE.

**PTF.** Program temporary fix.

\* **PU.** Physical unit.

\* **punch card.** A card into which hole patterns can be punched; normally, it is characterized by 80 columns and 12 rows of punch positions.

\* **queue file.** A direct access file maintained by VSE/POWER that holds control information for the spooling of job input and job output.

**queue record.** A record in the queue file containing descriptive information about a job or job output.

\* **random processing.** The treatment of data without respect to its location on disk storage, and in an arbitrary sequence governed by the input against which it is to be processed.

**real address.** The address of a location in processor storage.

\* **real address area.** The area of virtual storage where virtual addresses are equal to real addresses.

\* **real address space.** The address space whose addresses map one-to-one to the addresses in processor storage.

**real mode.** A processing mode in which a program may not be paged. Contrast with *virtual mode*.

**real storage.** See *processor storage*.

**record.** A set of related data or words, treated as a unit. See *logical record*, *physical record*.

**recovery management support (RMS).** System routines that gather information about hardware failures and that initiate a retry of an operation that failed because of processor, I/O device, or channel errors.

\* **reentrant.** The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by several tasks.

**refresh release.** An upgraded VSE system with the latest level of maintenance for a release.

\* **relocatable module.** A library member of the type object. It consists of one or more control sections cataloged as one member.

\* **relocating loader.** A function that modifies addresses of a phase, if necessary, and loads the phase for running into the partition selected by the user.

\* **remote job entry (RJE).** Submission of jobs through an input unit that has access to a computer through a data link.

**residency mode (RMODE).** A program attribute that refers to the location where a program is expected to reside in virtual storage. RMODE 24 indicates that the program must reside in the 24-bit addressable area (below 16 megabytes), RMODE ANY indicates that the program can reside anywhere in 31-bit addressable storage (above or below 16 megabytes).

\* **restore.** To write back onto disk data that was previously written from disk onto an intermediate storage medium such as tape.

**RJE.** Remote job entry.

**RJE workstation.** Any workstation that is used for remote job submission and for the remote retrieval of output. See also *programmable* and *nonprogrammable workstation*.

**RMODE.** Residency mode.

**RMS.** Recovery management support.

\* **routine.** A program, or part of a program, that may have some general or frequent use. (T)

\* **routing.** The assignment of the path by which a message will reach its destination.

\* **RPG II.** A commercially oriented programming language specifically designed for writing application programs intended for business data processing.

**SAM.** Sequential access method.

**SAM ESDS file.** A SAM file managed in VSE/VSAM space, so it can be accessed by both SAM and VSE/VSAM macros.

**schedule.** To select a program or task for getting control over the processor.

**SCP.** System control programming.

**SDL.** System directory list.

\* **SDLC.** Synchronous Data Link Control.

\* **search chain.** The order in which chained sublibraries are searched for the retrieval of a certain library member of a specified type.

**second-level directory.** A table in the SVA containing the highest phase names found on the directory tracks of the system sublibrary.

**security.** See *access control*.

\* **selection panel.** A displayed list of items from which a user can make a selection. Synonymous with *menu*.

**sequential access.** The serial retrieval of records in their entry sequence or serial storage of records with or without a premeditated order. Contrast with *direct access*.

**sequential access method (SAM).** A data access method that writes to and reads from an I/O device record after record (or block after block). On request, the support performs device control operations such as line spacing or page ejects on a printer or skip a certain number of tape marks on a tape drive.

\* **sequential file.** A file in which records are processed in the order in which they are entered and stored.

\* **service program.** A computer program that performs functions in support of the system. Synonymous with *utility program*.

**shared area.** An area of storage that is common to all address spaces in the system. VSE/ESA has two shared areas:

1. The shared area (24 bit) is allocated at the start of the address space and contains the supervisor, the SVA (for system programs and the system GETVIS area), and the shared partitions.
2. The shared area (31 bit) is allocated at the end of the address space and contains the SVA (31 bit) for system programs and the system GETVIS area.

**shared disk option.** An option that lets independent computer systems use common data on shared disk devices.

\* **shared partition.** A partition allocated for a program such as VSE/POWER that provides services for and communicates with programs in other partitions of the system's virtual address spaces.

\* **shared spooling.** A function that permits the VSE/POWER account file, data file, and queue file to be shared among several computer systems with VSE/POWER.

\* **shared virtual area (SVA).** A high address area that contains a system directory list (SDL) of frequently used phases, resident programs that can be shared between partitions, and an area for system support.

\* **skeleton.** A set of control statements, instructions, or both, that requires user-specific information to be inserted before it can be submitted for processing.

**SNA.** Systems Network Architecture.

\* **SNA network.** The part of a user-application network that conforms to the formats and protocols of SNA.

\* **software.** All or part of the programs, procedures, rules, and associated documentation of a data processing system. Software is an intellectual creation that is independent of the medium on which it is recorded. (T)

**source member.** A library member containing source statements in any of the programming languages supported by VSE.

\* **source program.** A program that a particular translator can accept. (T) Contrast with *object module*.

\* **source statement.** A statement written in symbols of a programming language.

**spanned record.** A record that extends over several blocks.

\* **spooling.** The use of disk storage as buffer storage to reduce processing delays when transferring data between peripheral equipment and the processors of a computer. In VSE, this is done under the control of VSE/POWER.

**SQL/DS™.** Structured Query Language/Data System.

**SS system.** Start-stop system.

**stand-alone program.** A program that runs independently of (not controlled by) the VSE system.

\* **standard label.** A fixed-format record that identifies a volume of data such as a tape reel or a file that is part of a volume of data.

\* **start-stop (SS) system.** A data transmission system in which each character is preceded by a start signal and is followed by a stop signal. (T)

**startup.** The process of performing IPL of the operating system and of getting all subsystems and application programs ready for operation.

**static partition.** A partition, defined at IPL time and occupying a defined amount of virtual storage that remains constant. Contrast with *dynamic partition*.

**station.** 1. One of the input or output points of a network that uses communication facilities; for example, the telephone set in the telephone system or the point where the business machine interfaces with the channel on a leased private line. 2. One or more computers, terminals, or devices at a particular location.

**storage dump.** See *dump*.

**storage fragmentation.** Inability to allocate unused sections (fragments) of storage in the real or virtual address range of virtual storage.

**Structured Query Language/Data System.** An IBM licensed program for using a database in an online, interactive, or batch environment.

\* **suballocated file.** A VSE/VSAM file that occupies a portion of an already defined data space. The data space may contain other files. Contrast with *unique file*.

\* **subarea.** A portion of the SNA network consisting of a subarea node, attached peripheral nodes, and associated resources. Within a subarea node, all NAUs, links, and adjacent link stations in attached peripheral or subarea nodes that are addressable within the subarea share a common subarea address and have distinct element addresses.

**sublibrary.** A subdivision of a library. Members can only be accessed in a sublibrary.

**sublibrary directory.** An index for the system to locate a member in the accessed sublibrary.

**submit.** A VSE/POWER function that passes a job to the system for processing.

\* **subsystem.** A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system. (T)

**subtask.** A task that is initiated by the main task or by another subtask.

\* **supervisor.** The part of a control program that coordinates the use of resources and maintains the flow of processor operations.

**SVA.** Shared virtual area.

\* **switched line.** A telecommunication line in which the connection is established by dialing.

**Synchronous Data Link Control (SDLC).** A discipline for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges may be duplex or half-duplex over switched or non-switched links. The configuration of the link connection may be point-to-point, multipoint, or loop.

**SYSRES.** System residence file.

\* **system console.** A console, usually equipped with a keyboard and display screen for control and communication with the system.

**system control programming (SCP).** IBM supplied, nonlicensed program fundamental to the operation of a system or to its service or both.

**system directory list (SDL).** A list containing directory entries of frequently-used phases and of all phases resident in the SVA. The list resides in the SVA.

\* **system file.** A file used by the operating system, for example, the hardcopy file, the recorder file, the page data set.

**system logical unit.** A logical unit available primarily for operating system use. See also *logical unit name*.

\* **system recorder file.** The file used to record hardware reliability data. Synonymous with *recorder file*.

**system refresh release.** See *refresh release*.

**system residence file (SYSRES).** The system sublibrary IJSYSRS.SYSLIB that contains the operating system. It is stored on the system residence volume DOSRES.

**system sublibrary.** The sublibrary that contains the operating system. It is stored on the system residence volume (DOSRES).

**Systems Network Architecture (SNA).** The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

\* **tailor.** A process that defines or modifies the characteristics of the system.

**task management.** The functions of a control program that control the use by tasks of the processor and other resources (except for input/output devices).

\* **telecommunication.** Transmission of data between computer systems over telecommunication lines and between a computer system and remote devices.

\* **telecommunication line.** Any physical medium such as a wire or microwave beam, that is used to transmit data. Contrast with *data link*.

\* **terminal.** A point in a system or network at which data can either enter or leave. (A) Usually a display screen with a keyboard.

\* **throughput.** 1. A measure of the amount of work performed by a computer system over a given period

of time, for example, number of jobs per day. (I) (A) 2. In data communication, the total traffic between stations per unit of time.

\* **track hold.** A function that protects a track that is being updated by one program from being accessed by another program.

\* **transient area.** An area within the control program used to provide high-priority system services on demand.

**transmission line.** Synonym for *telecommunication line*.

\* **transmit.** To send data from one place for reception elsewhere. (A)

**UCB.** Universal character set buffer.

**UCS.** Universal character set.

\* **unattended mode.** A mode in which no operator is present or in which no operator station is included at system generation.

**unattended node support.** A set of functions allowing one or more VSE systems to run without an operator being present. The systems are connected to a single central host.

\* **unique file.** A VSE/VSAM file that occupies a data space of its own. The data space is defined at the same time as the file and cannot contain any other file.

Contrast with *suballocated file*.

\* **universal character set (UCS).** A printer feature that permits the use of a variety of character arrays.

**universal character set buffer (UCB).** A buffer to hold UCS information.

**user console.** In VSE/ESA, a console that receives only those system messages that are specifically directed to it. These are, for example, messages that are issued from a job that was submitted with the request to echo its messages to that console. Contrast with *master console*.

\* **utility program.** 1. A computer program in general support of computer processes; for example, a diagnostic program, a trace program, or a sort program. (T) Synonymous with *service program*. 2. A program designed to perform an everyday task such as copying data from one storage device to another. (A)

**VAE.** Virtual addressability extension.

**virtual address.** An address that refers to a location in virtual storage. It is translated by the system to a processor storage address when the information stored at the virtual address is to be used.

**virtual addressability extension (VAE).** A storage management support that gives the user of VSE/ESA multiple address spaces of virtual storage.

**virtual address area.** The virtual range of available program addresses.

\* **virtual address space.** A subdivision of the virtual address area available to the user for the allocation of private, nonshared partitions.

**virtual disk.** A range of up to two gigabytes of contiguous virtual storage addresses that a program can use as workspace. Although the virtual disk exists in storage, it appears as a real FBA disk device to the user program. All I/O operations directed to a virtual disk are intercepted and the data to be written to, or read from, the disk is moved to or from a data space.

Like a data space, a virtual disk can hold only user data; it does not contain shared areas, system data or programs. Unlike an address space or a data space, data is not directly addressable on a virtual disk. To manipulate data on a virtual disk, the program has to perform I/O operations.

**VIO.** Virtual I/O area.

\* **virtual I/O area (VIO).** An extension of the page data set used by the system as intermediate storage, primarily for control data.

\* **virtual machine (VM).** A functional simulation of a computer system and its associated devices.

\* **virtual mode.** The operating mode of a program which may be paged.

\* **virtual partition.** A division of the dynamic area of virtual storage.

**virtual storage.** Addressable space image for the user from which instructions and data are mapped into processor (real) storage locations. :gt.VMESA mode :gd.An operation mode of the supervisor (generated with MODE=VMESA) of a VSE system running as a guest system under a VM operating system. :gt.VM mode :gd.An operation mode of the supervisor (generated with MODE=VM) of a VSE system running as a virtual machine under a VM operating system.

**volume.** A data carrier that is mounted and demounted as a unit, for example, a reel of tape or a disk pack. (I) Some disk units have no demountable packs. In that case, a volume is the portion available to one read/write mechanism.

**volume ID.** The volume serial number, which is a number in a volume label assigned when a volume is prepared for use by the system.

**volume table of contents (VTOC).** A table on a disk volume that describes every file on it. Acronym = VIRTUAL STORAGE EXTENDED

**VSE (Virtual Storage Extended).** A system that consists of a basic operating system (VSE/Advanced Functions) and any IBM supplied and user-written programs required to meet the data processing needs of a user. VSE and the hardware it controls form a complete computing system. Its current version is called VSE/ESA.

**VSE/Advanced Functions.** The basic operating-system component of VSE/ESA.

**VSE/DITTO (VSE/Data Interfile Transfer, Testing, and Operations Utility).** An IBM licensed program that provides file-to-file services for disk, tape, and card devices.

**VSE/ESA (VSE/Enterprise Systems Architecture).** The most advanced VSE system currently available.

\* **VSE/FCOPY (VSE/Fast Copy Data Set program).** An IBM licensed program for fast copy data operations from disk to disk and dump/restore operations via an intermediate dump file on magnetic tape or disk. There is also a stand-alone version: the FASTCOPY utility.

\* **VSE/ICCF (VSE/Interactive Computing and Control Facility).** An IBM licensed program that serves as interface, on a time-slice basis, to authorized users of terminals linked to the system's processor.



**VSE/ICCF library.** A file composed of smaller files (libraries) including system and user data which can be accessed under the control of VSE/ICCF.

**VSE library.** A collection of programs in various forms and storage dumps stored on disk. The form of a program is indicated by its member type such as source code, object module, phase, or procedure. A VSE library consists of at least one sublibrary which can contain any type of member. Acronym = OPERATOR COMMUNICATION CONTROL FACILITY

**\* VSE/OLTEP (VSE/Online Test Executive Program).**

An IBM program for managing the online tests that are available for preventive service for I/O devices. Normally, only IBM service representatives use this program.

**\* VSE/POWER.** An IBM licensed program primarily used to spool input and output. The program's networking functions enable a VSE system to exchange files with or run jobs on another remote processor.

**VSE/SP Unique Code.** A component of VSE/ESA.

**VSE/VSAM (VSE/Virtual Storage Access Method).**

An IBM access method for direct or sequential processing of fixed and variable length records on disk devices.

**VSE/VSAM catalog.** A file containing extensive file and volume information that VSE/VSAM requires to locate files, to allocate and deallocate storage space, to verify the authorization of a program or an operator to gain access to a file, and to accumulate use statistics for files.

**\* VSE/VSAM managed space.** A user-defined space on disk placed under the control of VSE/VSAM.

**VTAM (Virtual Telecommunications Access Method).**

An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability; it supports application programs and subsystems (VSE/POWER, for example).

**VTOC.** Volume table of contents.

**wait state.** The condition of a processor when all operations are suspended. System recovery from a hard wait is impossible without performing a new system startup. Synonym for *hard wait*.

**workstation.** See *programmable* and *nonprogrammable workstation*.

**Workstation File Transfer Support.** Enables the exchange of data between IBM Personal Computers (PCs) linked to a VSE/ESA host system where the data is kept in intermediate storage. PC users can retrieve that data and work with it independently of VSE/ESA. :gt.370 mode :gd.An operation mode of the supervisor (generated with MODE=370) of a VSE system. Such a supervisor supports multiple virtual address spaces and requires a processor of the System/370 and /390 architecture.

**31-bit addressing.** Provides addressability for address spaces of up to 2 gigabytes. (The maximum amount of addressable storage in previous systems was 16 megabytes.)

**370-XA.** IBM System/370 Extended Architecture.



---

## Index

### Special characters

: (colon) 372  
?? (question marks) 372  
// (control statement identification) 48  
// DLBL statement 395  
// EXTENT statement 426  
/. (label) command/statement 252  
/. label statement (librarian) 341  
/\$ (end-of-data) indicator 425  
/\* (end-of-data file) statement 254  
/\* (librarian end-of-session) 343  
/& (end-of-job) statement 255  
/+ (end-of-procedure) statement 253  
/+ (librarian end-of-data) 342  
(AMODE) addressing mode 267  
(RMODE) residence mode 267  
) ADD statement (ESERV) 348  
) COL statement (ESERV) 349  
) DEL statement (ESERV) 350  
) END statement (ESERV) 351  
) REP statement (ESERV) 352  
) RST statement (ESERV) 353  
) VER statement (ESERV) 354  
)ADD subcommand (UPDATE) 337  
)DEL subcommand (UPDATE) 338  
)END subcommand (UPDATE) 339  
)REP subcommand (UPDATE) 340  
\$ABEND operand 161  
\$CANCEL operand 161  
\$JOBEXIT 123  
\$MRC operand 117  
\$RC operand 117, 161, 319  
\* (comments) statement 257  
\* CP command 256  
| sign 371  
+ (plus) sign 372  
+rel operand  
    DELETE 428  
    INSERT 434  
    REPLACE 438  
    RESTART 442  
    VERIFY 445  
= (equal) sign 372  
' (single quotes) 372  
'file-id' operand  
    DLBL statement 86  
    TLBL statement 236  
'from' sublibrary, defining 300  
'to' sublibrary, defining 300

### Numerics

1403U printer 241  
24-bit SVA 32  
31-bit SVA 33  
3480 tape subsystem, assigning 62  
3494 tape device support (LIBSERV) 131  
3590 Magstar Tape unit  
    mode settings 68

3800 printing subsystem  
    controlling 214  
    default settings 208  
    defaults, overriding 214  
    device-type codes 65  
3800 settings, example 218  
3990-3 / 3990-6  
    cache fast write 72  
    DASD fast write 72  
    dual copy support 72  
4248 printer 125  
    assigning 62, 65  
4248 Printer 355

## A

abbreviation of librarian commands 287  
abbreviations of keywords 371  
ABEND condition 161  
abnormal conditions, ignoring 119  
ACANCEL option 167, 225  
    suppression by LOG 143  
ACCESS command (librarian) 290  
access rights, sublibrary 128  
accounting information (JOB statement) 124  
ACL option 167, 225  
ACTION statement (linkage editor) 273  
ADD command (IPL) 13  
adding member lines 337  
address operand (ALTER) 422  
ADDRESS operand (PERSONALIZE) 405  
address space 58  
address space layout 38  
addressing mode  
    assigning via MODE statement 278  
addressing mode (AMODE) 267  
AFFECTS detail control statement 420  
after-line operand (INSERT) 434  
ALIGN option 168, 225  
ALLOC command (attention routine) 56  
ALLOC command (job control) 56  
allocating storage 38  
    multiple-partition allocation 57  
    single-partition allocation 56  
alphanumeric characters, definition of 4  
ALT operand 67, 83  
ALTER command (attention routine) 59  
ALTER statement 422  
altering a phase or module 422  
altering mode of operation 165  
altering virtual storage 59  
alternate assignment (tape) 67  
alternate history file 368  
alternative options 371  
alternative specifications 371  
AMODE (addressing mode)  
    assigning via MODE statement 278  
AND relation (vs. OR relation) 436, 439  
APAR 373  
    archiving 376  
    listing 413

APAR (*continued*)  
   removing 407  
 APAR fix  
   installing 382  
   removing 419  
 apar-number operand 373  
   CORRECT 384  
   TAILOR 419  
 apar-number specification 373  
 APAR=apar-number operand 373  
   ARCHIVE 378  
   LOOKUP 399  
   REMOVE 407  
 APARS operand (RETRACE) 413  
 APARS=apar-number operand (RESOLVES) 441  
 applications  
   a single PTF 374  
   corrective service 396  
   preventive service 396  
 APPLY statement 374  
 applying 373  
 AR (attention routine), overview 1  
 ARCHIVE statement 376  
 archiving products/components/APARs/PTFs 376  
 ARGUMENT operand (SCAN) 443  
 ASI (automated system initialization) 7, 178, 205, 221, 246  
 assembler 365, 368, 430  
 ASSGN command/statement (job control) 60  
 ASSGN statement 365  
 ASSGN, resetting 201  
 assigning  
   addressing mode (AMODE) 278  
   residence mode (RMODE) 278  
 assigning devices 157  
 assigning logical units 60  
 assignment  
   altering 157  
   device 60  
   holding 115  
   listing 141  
   resetting 201  
   temporary v. permanent 60, 67  
 assignment of logical units 365  
 associative field 362  
 attention routine (AR), overview 1  
 attention routine commands  
   ALLOC 56  
   ALTER 59  
   BANDID 70  
   BATCH 71  
   CACHE 72  
   CANCEL 80  
   continuation of 5  
   DSPLY 91  
   DUMP 92  
   END 97  
   ENTER 97  
   FREE 110  
   GETVIS 111  
   IGNORE 119  
   IXFP 120  
   LFCB 125  
   LIBSERV 131  
   LUCB 145  
   MAP 146  
   MSECS 153  
   MSG 154  
   NEWVOL 157  
   NOLOG 158  
   OFFLINE 160  
   ONLINE 164  
   OPERATE 165  
   overview 47  
   PAUSE 176  
   PRTY 178  
   PRTYIO 182  
   PWROFF 185  
   QUERY 186  
   RC 194  
   REDISPLAY 195  
   REPLID 199  
   RESERV 200  
   SETDF 208  
   SIZE 219  
   START 221  
   STATUS 222  
   TPBAL 240  
   UNBATCH 242  
   UNLOCK 243  
   VOLUME 247  
 attention routine, interrupting 194  
 attention routine, retrieving 195  
 AUTOLINK (automatic library lookup) function 273, 282  
 automated system initialization (ASI) 7  
 automatic library lookup (AUTOLINK) function 273, 282  
 automatic print buffer loading 359  
 Automatic Volume Recognition (AVR) 164  
 auxiliary history file 368, 373  
   backing up 378  
   copying to disk 381  
   copying to tape 378  
   creating 385  
   defining 426  
   dumping 386  
   initializing 385  
 AUXILIARY operand 373  
   BACKUP HISTORY 379  
   COPY HISTORY 382  
   CREATE HISTORY 386  
   DEFINE HISTORY 426  
   DUMP HISTORY 387  
   MERGE HISTORY 402  
   RESTORE HISTORY 411  
 AVR (Automatic Volume Recognition) 164

## B

background partition, default 58  
 backout job 394  
 BACKOUT operand (INSTALL BACKOUT) 394  
 backout PTF 375  
   installing 392, 394  
 backout tape 392, 394  
 backspacing tape 156  
 backup  
   file header 294  
   file-ID 294, 328  
   history file 294  
   tape format 293  
 BACKUP command (librarian) 292  
 backup copy  
   of history file 378  
   of product 379

BACKUP HISTORY statement 378  
 BACKUP PRODUCT statement 379  
 backup tape  
   header file 381  
 balancing partitions 153, 179  
 band-id operand 355  
 band-id operand (4248) 70  
 BANDID command (attention routine) 70  
 BANDID statement (SYSBUFLD) 355  
 base product 389  
 Basic Security Manager (BSM) 35  
 BATCH command (attention routine) 71  
 BG partition, default 58  
 BKEND statement 296  
 blanks (as separators) 372  
 BLKSIZE operand 88  
 BLOCK operand 241  
 blocksize, calculation of 88  
 braces 371  
 brackets 371  
 BSF operand 156  
 BSM (Basic Security Manager) 35  
 BSR operand 156  
 buffer  
   data (VSE/VSAM) 88  
   index (VSE/VSAM) 88  
   load 125, 145  
   load phases, non-standard 357  
   load phases, standard 357  
   loading, automatic 359  
   phase names 355  
   space for VSAM files 88  
 BUFND operand 88  
 BUFNI operand 88  
 BUFSP operand 88  
 BURST operand 208, 214  
 bursting (3800 printer) 208, 214

**C**

CACHE command (attention routine) 72  
 cache fast write 72  
 cache settings 74  
 cache support 72  
 called system control programs 368  
 calling programs 98  
 CANCEL command (job control/attention routine) 80  
 cancel condition 161  
 CANCEL operand 274  
 canceling jobs or I/O requests 80  
 canceling MSHP control statements 372  
 card image input record 371  
 cartridge (IBM 3480, 3490) 64  
 CAT operand 88  
 CATAL option 168  
 CATALOG (Librarian) statement 425  
 CATALOG command (librarian) 295  
 CATALOG operand 127, 129  
 cataloged procedures 177  
 cataloging LIBDEF, restriction 127  
 cataloging phases 168, 280  
 chaining phases 128  
 CHANGE command (librarian) 297  
 changing a phase (PATCH) 403  
 channel program 74  
 channel queue entries, allocating 34  
 channel-to-channel adapter (CTCA) 13  
 character arrangement table (3800 printer) 208, 215  
 character-set option 174  
 CHARS operand 209, 215  
 CHARSET option 225  
 checkpoint, restarting from 203  
 CHKPT macro 203  
 CI (control interval) size 89  
 CISIZE operand 89  
 class, dynamic  
   MAP output 147, 148  
 CLASSTD option 168  
 CLOSE command/statement (job control) 60, 82  
 closing logical units 82  
 CM (ESD type) 450  
 CMD operand 165  
 CO operand (REQUIRES) 439  
 coding conventions 371, 372  
 colon (:): 372  
 COLUMN operand 335  
 command notation explained 2  
 command symbols 2  
 command-line (EXEC LIBR) 289  
 commas (as separators) 372  
 comment (RESOLVES statement) 441  
 comment input line 372  
 comment operand 441  
 comments (as separators) 372  
 comments statement 257  
 common storage 450  
 communication device, specifying 9  
 communications routine 154  
 comparator operand  
   IF command/statement 117  
   ON command/statement 161  
 COMPARE command (librarian) 298  
 compatibility considerations 396  
   APPLY statement 375  
   CORRECT statement 384  
   INSTALL SERVICE FROMTAPE statement 394  
   LIST FROMTAPE statement 396  
 compatible products 389, 423  
 COMPATIBLE statement 423  
 compile, link and go 101  
 compiler messages, displaying 227  
 component 373  
   archiving 376  
   correcting 382  
   installing in SYSIN format 388  
   listing (RETRACE) 413  
   removing from history file 406  
 component operand  
   COMPRISES 424  
   REQUIRES 439  
 component specification 372  
 component-level operand  
   ARCHIVE 378  
   REMOVE 406  
 component[-level] operand 373  
   APPLY 374  
   CORRECT 384  
   INCORPORATE 388  
   LOOKUP 398  
   REVOKE 414  
   TAILOR 417  
   UNDO 419  
 COMPONENT=component[-level] operand 373  
   EXCLUDE 429

COMPONENT=component[-level] operand (*continued*)

- INCLUDE 432
- SELECT 415
- COMPONENTS operand (RETRACE) 413
- comprised components 424
- COMPRISES statement 424
- concatenation of symbolic parameters 52
- conditional execution
  - GOTO command/statement 113
  - IF command/statement 117
  - ON command/statement 161
- conditional execution (librarian) 319
- conditional job control 50
- CONN operand 165
- CONNECT command (librarian) 300
- CONS operand 15
- console input 371
- console type, specifying 9
- console, unlocking 176
- continuation character 5
- continuation dash (-) 372
- continuation line 372
- continuation lines 5
- continuation of control statements 372
- CONTINUE operand 162, 319
- continuing checkpointed programs 203
- CONTINUOUS operand (LIST) 397
- control interval (CI) size 89
- control statement overview 366
- conventions, command 2
- COPIES operand 215
- COPY command (librarian) 301
- COPY HISTORY statement 381
- copy modification (3800 printer) 208, 216
- copying a history file 381
- corequisite 373, 436
  - condition 439
- CORRECT statement 382
- correcting a component 382
- corrective service, applying 396
- cover letter, PTF 397
- COVER operand (LIST) 397
- CREATE HISTORY statement 385
- creating a history file 385
- creating FCB image phases 359
- creating the recorder file 205
- creating UCB image phases 359
- creation date 236
- cross-reference list
  - of included macros 430
  - of PTFs, APARs 397
- cross-system communication file
  - defining 22
  - size of 23
- CTCA (channel-to-channel adapter) 13
- customer-name operand (PERSONALIZE) 404
- cuu, definition of 4, 13
- cylinder sizes in tracks 108

## D

- DA operand 87
- DASD fast write 72
- DASD file protection, activating 34
- DASD sharing 14
- dash (-) 372
- data buffers (VSE/VSAM) 88

- data check, ignoring 241
- DATA operand (LOOKUP) 400
- data protection, activating 34
- data space
  - defining (SYSDEF command/statement) 229
  - displaying information on 186
  - dump option 168
  - dumping 92
  - querying (QUERY command/statement) 186
- DATA statement 384, 425
- data-secured file 88
- DATE format option 225
- date operand
  - DLBL statement 87
  - TLBL statement 236
- DATE operand 302, 317
  - IPL SET command 26
- DATE statement (job control) 85
- date, setting 26
- DB2 data base 28
- DB2 guest sharing 28
- DCHK operand 215
- de-edited macros
  - cataloging 346
  - displaying 347
  - displaying and punching 347
  - punching 347
- de-editing macros 2, 345
- DEBUG operand 215
- DECK option 168, 226
- DEF command (IPL) 16
- DEF SCSI command (IPL) 17
- default ON conditions 162
- default symbolic parameters 177
- default values 365, 371
- defaults, overriding 167
- DEFINE command (librarian) 305
- DEFINE HISTORY statement 426
- defining a history file 426
- defining libraries 305
- defining SCSI devices 232
- defining sublibraries 305
- DEL command (IPL) 18
- DELETE command (librarian) 307
- DELETE statement 428
- deleting
  - I/O devices 18
  - libraries 307
  - member lines 338
  - members 307
  - partitions 58
  - sublibraries 307
- deleting lines from a macro 428
- detail control statements 365, 420
  - AFFECTS 420
  - ALTER 422
  - COMPATIBLE 423
  - COMPRISES 424
  - DATA 425
  - DEFINE HISTORY 426
  - DELETE 428
  - EXCLUDE 429
  - EXECUTE 430
  - for ARCHIVE 376
  - GENERATE 431
  - INCLUDE 432
  - INFLUENCES 433

- detail control statements (*continued*)
  - INSERT 434
  - INVOLVES 435
  - optional 365
  - OR 436
  - overview 367
  - PTF 436
  - REPLACE 438
  - required 365
  - REQUIRES 439
  - RESOLVES 441
  - RESTART 442
  - SCAN 443
  - sequence of 371
  - SUPERSEDES 444
  - VERIFY 445
- DEV command (IPL) 19
- device
  - classes 64
  - emulation 14
  - sensing, ignored 14
  - types 40
- device assignment
  - altering 157
  - holding 115
  - listing 141
  - resetting 201
  - temporary v. permanent 60, 67
- device number, definition of 4
- device status, displaying 222
- device type codes (IPL) 40
- device-class operand
  - ASSGN 63
  - CLOSE 84
- device-down (DVCDN) command 95
- device-not-ready status, simulating 160
- device-ready status, simulating 164
- device-up (DVCUP) command 96
- DFLT operand 215
- dictionary record 269
- directories, listing 311
- DISC operand 165
- disconnected console 15
- disk label information, defining (DLBL) 86
- disk labels, defining (DLBL) 86
- disk sharing 14
- diskette extents, defining 106
- diskette labels, defining (DLBL) 86
- DISP operand
  - DLBL statement 89
  - TLBL statement 238
  - VSE/VSAM disk files 89
- displaying 396
  - APAR information 399
  - component information 398
  - history file 397
  - I/O devices 19
  - macro information 400
  - module information 399
  - phase information 399
  - product information 398
  - PTF information 399
- displaying reply-IDs 199
- displaying SCSI related information 191
- displaying status of multiprocessor environment 192
- displaying the GETVIS area 111
- displaying virtual storage 91

- disposition
  - tape files 238
  - VSE/VSAM disk files 89
- distribution 373
  - libraries 389
  - tape 390
- DLA command (IPL) 20
- DLBL and EXTENT sequence 50
- DLBL statement 395, 396
  - DEFINE HISTORY 426
  - LIST FROMDISK 396
- DLBL statement (job control) 86
- DLF command (IPL) 22
- DOCUMENT operand (LIST) 396
- documentation, service tape 396
- DPD command (IPL) 24
- dropping search chains 129
- DSE operand 156
- DSF operand 88
- DSPCH statement (ESERV) 347
- DSPDUMP option 168, 226
- DSPLY command (attention routine) 91
- DSPLY statement (ESERV) 347
- DU operand 87
- dual copy support 72
- DUMP command (attention routine) 92
- DUMP HISTORY statement 386
- DUMP operand 81
- DUMP option 169, 226
- dump, stand-alone 171
- dumping data spaces 92
- dumping the history file 386
- dumping virtual storage 92
- dumps of data spaces 92
- dumps of virtual storage 92
- dumps, suppressing of 169
- dumps, types of 169
- DVCDN command (job control) 95
- DVCUP command (job control) 96
- dynamic class
  - MAP output 147, 148
  - standard label group 168
- dynamic partition
  - altering storage 59
  - assignments, listing of 141
  - device assignments, listing of 141
  - displaying storage 91
  - dumping 93
  - GETVIS area, displaying 111
  - I/O device assignments, listing of 141
  - I/O request priority setting 182
  - listing I/O assignments 141
  - logging job control statements 144
  - maximum number of 35
  - priority setting 178

## E

- ECKD device-type 64
- edited macro service program (ESERV) 345
  - overview 2
- editing macros 2, 345
- ellipsis 372
- emulation 14
- END statement (librarian) 343
- END statement (linkage editor) 449
- end-of-data (/) indicator 425

end-of-data delimiter (librarian) 296  
 end-of-data file statement (/\*) 254  
 end-of-job statement (/&) 255  
 end-of-procedure statement (/+) 253  
 end-of-session (/\*) statement (librarian) 343  
 END/ENTER command 97  
 entry point 275  
 ENTRY statement (linkage editor) 275  
 ENVIRONMENT operand (PERSONALIZE) 405  
 EOD operand 296  
 EOF operand 322  
 equal (=) sign 372  
 ER (ESD type) 450  
 erasing tape 156  
 erasing tape gap 156  
 ERG operand 156  
 ERRLMT operand 274  
 ERRS option 169, 226  
 ESD (external symbol dictionary) 269, 450  
 ESD (external symbol dictionary) statement 447  
 ESDID= operand (AFFECTS) 420  
 ESERV  
   program name (ESERV) 345  
 ESERV (edited macro service program) 345  
   overview 2  
 ESERV (edited macro service program) control statements  
   ) ADD 348  
   ) COL 349  
   ) DEL 350  
   ) END 351  
   ) REP 352  
   ) RST 353  
   ) VER 354  
   DSPCH 347  
   DSPLY 347  
   GENCATALS 346  
   overview 345  
   PUNCH 347  
 ESERV (program name for ESERV) 345  
 ESM (External Security Manager) 35  
 examples 373  
   AFFECTS 421  
   ALTER 422  
   APPLY 375  
   ARCHIVE 378  
   ASSGN statement 258  
   BACKUP HISTORY 379  
   BACKUP PRODUCT 381  
   COMPATIBLE 423  
   COMPRISES 424  
   COPY HISTORY 382  
   CORRECT 384  
   CREATE HISTORY 386  
   DATA 425  
   DEFINE HISTORY 427  
   DELETE 428  
   DLBL statement 258  
   DUMP HISTORY 387  
   ESERV, invoking 345  
   EXCLUDE 429  
   EXEC statement 258  
   EXECUTE 430  
   EXTENT statement 258  
   FCB image phases 363  
   GENERATE 431  
   GOTO statement 262  
   IF statement 261  
   examples (*continued*)  
     image phases (FCB) 363  
     INCLUDE 432  
     INCORPORATE 388  
     INFLUENCES 433  
     INSERT 434  
     INSTALL PRODUCT/SYSRES 392  
     INSTALL SERVICE/BACKOUT 395  
     invoking ESERV 345  
     INVOLVES 435  
     job control (general) 258  
     JOB statement 258  
     LIBDEF statement 260  
     LIST SERVICETAPE 397  
     LOOKUP 400  
     member sequence numbering 295, 336  
     MERGE HISTORY 402  
     merging sublibraries 301, 315  
     MTC statement 258, 259  
     nested procedures 264  
     ON statement 262  
     OPTION CATAL statement 260  
     OPTION statement 259  
     OR 436  
     parameterized procedure 263  
     PATCH 403  
     PERSONALIZE 405  
     PROC statement 263  
     procedure nesting 264  
     PTF 437  
     REMOVE 407  
     REPLACE 438  
     REQUIRES 440  
     RESIDENCE 408  
     RESOLVES 441  
     RESTART 442  
     RESTORE HISTORY 411  
     RESTORE PRODUCT 410  
     RETRACE 413  
     REVOKE 414  
     SCAN 443  
     SELECT 415  
     sequence numbering 336  
     SETPARM statement 263  
     sublibrary merging 315  
     SUPERSEDES 444  
     symbolic parameters 263  
     TAILOR 418  
     VERIFY 445  
   EXCLUDE statement 429  
   excluding products, components, PTFs (from service) 429  
   excluding PTFs (from service) 392  
   EXEC MSHP control statement 365  
   EXEC PROC statement (job control) 51  
   EXEC statement/command 98  
   EXECUTE statement 430  
   executing a program 98  
   executing MSHP 365  
   executing system programs 430  
   EXPAND= operand (AFFECTS) 420  
   expanding a phase or module 420  
   expiration date 87  
   EXPLAIN statement 105  
   EXTENT and DLBL sequence 50  
   extent information 365  
   extent information (DEFINE HISTORY) 427  
   EXTENT operand (DEFINE HISTORY) 427



- extent size 109
- EXTENT statement 426
- EXTENT statement (job control) 105
- extent type 107
- extent-full (SYSLST) 204
- extent-full (SYSPCH) 204
- EXTents operand 306
- external reference 450
- External Security Manager (ESM) 35
- external symbol dictionary (ESD) 269, 450

## F

- FBA operand 62
- FCB (forms control buffer)
  - 3800 printer 208, 216
  - characters 362
  - image 361
  - image phase format 361
  - image phases 357
  - image phases, creating 359
  - image phases, example 363
  - loading via LFCB command 125
  - loading via SYSIPT 362
  - phase names 355
- FCB operand 209, 216
- FCB statement (SYSBUFLD) 356
- feature number 373, 375, 385
- file disposition
  - tape files 238
  - VSE/VSAM disk files 89
- file identifier 427
- file protection, activating 34
- file sequence number 108
- file-ID (backup tape) 328
- file-id operand
  - DLBL statement 86
  - TLBL statement 236
- file-section-number operand 238
- file-sequence-number operand 238
- file-serial-number operand 238
- file, data-secured 88
- file, multi-volume 108
- filename operand
  - DLBL statement 86
  - TLBL statement 236
- fix, installing 382
- FLASH operand 209, 216
- flashing (3800 printer) 208, 216
- FOLD operand 70, 145, 241, 355, 356
- font offset table 361
- FORCE operand 81, 161
- format of backup tape 293
- FORMAT=HEX operand 310
- FORMAT=IEBU<sub>p</sub>dte operand 322
- FORMAT=NOHEADER operand 322
- FORMAT=OLD operand 321
- FORMS operand 125, 209, 216
- forms overlay (3800 printer) 208
- forward space tape 156
- FREE command (attention routine) 110
- free form statements 371
- free space, releasing 323
- freeing library space 306
- freeing reserved status 110
- frequently used MSHP operands 372

- from-line operand
  - DELETE 428
  - REPLACE 438
- FROMDISK operand
  - INSTALL PRODUCT/SYSRES 390
  - INSTALL SERVICE 395
- FROMTAPE operand
  - INSTALL PRODUCT/SYSRES 390
  - INSTALL SERVICE 394
  - LIST 396
- FSF operand 156
- FSR operand 156
- function control statements 365
  - APPLY 374
  - ARCHIVE 376
  - BACKUP HISTORY 378
  - BACKUP PRODUCT 379
  - COPY HISTORY 381
  - CORRECT 382
  - CREATE HISTORY 385
  - DUMP HISTORY 386
  - INCORPORATE 388
  - INSTALL BACKOUT 392
  - INSTALL PRODUCT/SYSRES 388
  - INSTALL SERVICE 392
  - LIST 396
  - LOOKUP 397
  - MERGE HISTORY 401
  - overview 366
  - PATCH 403
  - PERSONALIZE 403
  - REMOVE 406
  - RESIDENCE 408
  - RESTORE HISTORY 411
  - RESTORE PRODUCT/SYSRES 409
  - RETRACE 412
  - REVOKE 414
  - SELECT 415
  - TAILOR 416
  - UNDO 419
- function of MSHP statements 365

## G

- GENCATALS statement (ESERV) 346
- GENERATE statement 431
- generating sublibrary members 416
- generation file 415, 430
- GENERATION INTO operand
  - INSTALL PRODUCT/SYSRES 392
  - RESTORE PRODUCT/SYSRES 410
- GENERATION operand (BACKUP PRODUCT) 381
- generation part (of shipment package) 392
- generation sublibrary 380, 381
- generation-number operand 238
- GENERATION=lib.sublib operand (RESIDENCE) 408
- generic operands (librarian) 288
- GENFILE operand (SELECT) 415
- GETVIS area
  - displaying 111
  - mapping 150
  - size 99, 100
  - size, altering 219
- GETVIS command (attention routine) 111
- GO operand 101
- GOTO command (librarian) 308
- GOTO command/statement (job control) 113

GOTO operand 162, 319  
GOTO statement example 262  
guest sharing (VM) 28

## H

hard-copy file  
  defining 16  
hardcopy file  
  creating 204  
  writing to 202  
HC operand 204  
HCLOG command (job control) 114  
HCTRAN option 226  
header file 381  
header, backup file 294  
HEADER=member-name operand (BACKUP PRODUCT) 381  
hexadecimal member list 310  
High Level Assembler for VSE 368  
history file 373  
  alternate 368  
  auxiliary 368  
  backing up 378  
  backing-up 294  
  copying from disk to disk 381  
  copying to tape 378  
  creating 385  
  defining 16, 426  
  dumping 386  
  identifying 403  
  in VSAM-managed space 369  
  initializing 385  
  merging 401  
  personalizing 403  
  printing information from 412  
  removing entries from 406  
  repairing the history file 370  
  restoring 411  
  restrictions 369  
  second 368  
  shipment 389  
  system 368  
  types of 368  
HOLD command (job control) 115  
holding assignments/sublibrary definitions 115  
horizontal copy function 125

## I

I/O assignments, listing 141  
I/O requests  
  canceling 80  
  setting priorities of 182  
IBM 1403U printer 241  
IBM 3480 tape subsystem, assigning 62  
IBM 3494 tape device support (LIBSERV) 131  
IBM 3800 printing subsystem  
  controlling 214  
  default settings 208  
  defaults, overriding 214  
  device-type codes 65  
IBM 3990-3 / 3990-6  
  cache fast write 72  
  DASD fast write 72  
  dual copy support 72  
IBM 4248 printer 125

IBM 4248 printer (*continued*)  
  assigning 62, 65  
IBM 4248 Printer 355  
ID command/statement 116  
ID operand 328  
ID='tapefile-id' operand  
  BACKUP PRODUCT 381  
  INSTALL PRODUCT/SYSRES 390  
  RESTORE PRODUCT/SYSRES 410  
IDENTIFIER= operand (DEFINE HISTORY) 427  
IDENTIFIER=component operand (RETRACE) 413  
IF command/statement 117  
IF statement example 261  
IGN operand 63, 83  
IGNORE command (JC/AR) 119  
ignoring abnormal conditions 119  
ignoring logical units 63, 83  
IJSYS02 work file 368  
IJSYSRn 391  
IJSYSRS.SYSLIB 127  
image phase formats (UCB, FCB) 361  
in use (of sublibraries) 285  
INCLUDE statement 432  
INCLUDE statement (linkage editor) 276  
including products, components, PTFs (in service) 432  
including PTFs (in service) 392  
INCORPORATE statement 388  
index buffers (VSE/VSAM) 88  
INDIRECT operand (APPLY) 375  
indirect PTF application 375  
INFLUENCES statement 433  
INIT operand 216  
initial program load (IPL), overview 1  
initializing symbolic parameters 177  
INPUT command (librarian) 309  
input line 371, 372  
INSERT statement 434  
inserting source input in a macro 434  
INSTALL BACKOUT statement 392  
INSTALL PRODUCT/SYSRES statement 388  
INSTALL SERVICE statement 392  
installing 373  
  APAR fix 382  
  backout PTFs 392, 394  
  component in SYSIN format 388  
  from disk 390  
  from tape 390  
  local fix 382  
  product 388  
  PTFs from service tape/file 392  
  single PTF 374  
  SYSRES 388  
interrupting attention routine 194  
INTO operand  
  INSTALL PRODUCT/SYSRES 391  
  RESTORE PRODUCT/SYSRES 410  
invalid JC commands and statements 50  
invoking the librarian 288  
INVOLVES statement 435  
IPL (initial program load) commands  
  ADD 13  
  DEF 16  
  DEF SCSI 17  
  DEL 18  
  DEV 19  
  DLA 20  
  DLF 22

IPL (initial program load) commands *(continued)*

- DPD 24
- overview 7
- SET 26
- SET XPCC 28
- SET ZONEBDY command 30
- SET ZONEDEF command 29
- supervisor parameters 11
- SVA 32
- SYS 34

IPL (initial program load), overview 1

IPL load parameter 9

IPL message suppression 9

IPL prompting 9

IRREVOKABLE operand

- APPLY 375
- CORRECT 384

ISC operand 87

ISE operand 87

IXFP command (JC/AR) 120

## J

JCANCEL option 169, 226

JCL exit routine 123

JCL source statement logging 170

JCLEXIT command 123

job

- accounting 124
- accounting, activating 34
- canceling 80
- continuing from checkpoint 203
- defining 124
- duration 255
- name 124
- restarting from checkpoint 203
- time 255

job control

- conditional 50
- example (general) 258
- options, standard 225
- program, overview 1

job control options

- ACANCEL 167, 225
- ACL 167, 225
- ALIGN 168, 225
- CATAL 168
- character-set 174
- CHARSET 225
- CLASSTD 168
- DATE 225
- DECK 168, 226
- DSPDUMP 168, 226
- DUMP 169, 226
- ERRS 169, 226
- HCTRAN 226
- JCANCEL 169, 226
- LINES 226
- LINK 169
- LIST 170, 226
- LISTX 170, 226
- LOG 143, 170, 226
- LOGSRC 54, 170
- NODUMP 169
- NOSLISKIP 172
- PARSTD 171
- PARTDUMP 169

job control options *(continued)*

- RLD 171, 226
- SADUMP 171, 226
- SCANCEL 172, 227
- SLISKIP 172
- STDLABEL 172
- SUBLIB=AE 173
- SUBLIB=DF 173
- SXREF 174, 227
- SYM 173, 227
- SYSDUMP 173, 227
- SYSDUMPC 173, 227
- SYSPARM 174
- TERM 174, 227
- USRLABEL 174
- XREF 174, 227

job control statements and commands

- /. (label) 252
- /\* (end-of-data file) 254
- /& (end-of-job) 255
- /+ (end-of-procedure) 253
- \* (comments) 257
- \* CP 256
- ALLOC 56
- ASSGN 60
- CANCEL 80
- CLOSE 60, 82
- comments 257
- conditional execution 50
- continuation of 5
- CP 256
- DATE 85
- DLBL 50, 86
- DVCDN 95
- DVCUP 96
- END 97
- end-of-data file (/\*) 254
- end-of-job (/&) 255
- end-of-procedure (/+) 253
- ENTER 97
- EXEC 98
- EXEC PROC 51
- EXTENT 50, 105
- GOTO 113
- HOLD 115
- ID 116
- IF 117
- IGNORE 119
- invalid 50
- IXFP 120
- JCLEXIT 123
- JOB 124
- label (/.) 252
- LIBDEF 126
- LIBDROP 129
- LIBLIST 130
- LIBSERV 131
- LISTIO 141
- LOG 143
- MAP 146
- MSECS 153
- MTC 155
- NOLOG 158
- NPGR 159
- ON 161
- operand delimiters 48
- OPTION 167

job control statements and commands (*continued*)

- overview 47
- parameterized 51
- PAUSE 176
- printing 54
- PROC 51, 177
- PRTY 178
- PWR 184
- QUERY 186
- RESET 201
- ROD 202
- RSTRT 203
- rules for coding 48
- sequence 50
- SET 204
- SETPARM 51, 210
- SETPFIX 212
- SETPRT 214
- SETPRT, example 218
- SIZE 219
- START 221
- STATUS 222
- STDOPT 225
- STOP 228
- summary 49
- syntax rules 48
- SYSECHO 235
- TLBL 236
- UCS 241
- UPSI 244
- VDISK 245
- VTAPE 249
- ZONE 251

job restart 394

JOB statement 124

job step 50

jobname operand 124

## K

KEEPDATA operand (TAILOR) 417

keyword abbreviation 371

keyword operands 288

keywords 371

## L

label

- adding 171
- definition 450
- deleting 171
- information, temporary 174
- operand (librarian) 308
- reference 450
- statement/command 113, 252
- subarea 171
- tape 236

label (/.) command/statement (job control) 252

label area

- clearing 20
- defining 20
- in virtual storage 246
- size of 21

label area in virtual storage 246

label information area 20

label operand 113

label statement (librarian) 341

labels, standard tape 379, 381, 390

LD/LR (ESD types) 450

level (of component) 373

level number (of component) 373

LFCB command 125

lib specification 373

LIBDEF command/statement 126

- restriction in procedure 127

LIBDEF, resetting 201

LIBDROP command/statement 129

LIBLIST command/statement 130

LIBR (program name of librarian) 289

librarian 365, 368, 430

- end-of-data 342
- end-of-session 343
- input from SYSIPT 309
- input, redirecting 309
- invoking 288
- LIBR (program name) 289
- migration (MAINT) 321
- output, redirecting 310, 313
- partition size 289
- program name (LIBR) 289
- punch output format 321
- return codes 285

librarian commands

- /. label 341
- /\* (end-of-session) 343
- /+ (end-of-data) 342

abbreviations 287

ACCESS 290

BACKUP 292

CATALOG 295

CHANGE 297

COMPARE 298

CONNECT 300

COPY 301

DEFINE 305

DELETE 307

END 343

end-of-data (/+) 342

end-of-session (/\*) 343

GOTO 308

INPUT 309

label 341

LIST 310

LISTDIR 311

LOCK 314

MOVE 315

ON 319

PUNCH 321

RELEASE 323

RENAME 324

RESTORE 325

SEARCH 330

summary 289

syntax 286

TEST 332

UNLOCK 333

UPDATE 334

UPDATE subcommands 336

librarian program, overview 1

libraries

- defining 305
- deleting 307
- restoring 325

- libraries (*continued*)
  - space, freeing 306
  - structure 285
- library name 373
- LIBSERV
  - AQUERY 135
  - CANCEL 135
  - CMOUNT 135
  - CQUERY 135
  - DQUERY 135
  - EJECT 135
  - IQUERY 136
  - LQUERY 136
  - MINVENT 136
  - MOUNT 136
  - PASS 137
  - RELEASE 137
  - SETVCAT 138
  - SQUERY 138
- LIBSERV command/statement 131
- licensed program 389
- line count, setting 204
- LINECT operand 204
- LINES option 226
- LINK operand (INVOLVES) 435
- LINK option 169
- link step 394
- link-book 435
- link-editing 435
- linkage editor 365, 368, 430
  - header line 268
  - invoking 267
  - overview 1
  - program name (LNKEDT) 267
  - return codes 269
- linkage editor control statements 425
  - ACTION 273
  - END 449
  - ENTRY 275
  - ESD 447
  - format 270
  - INCLUDE 276
  - MODE 278
  - PHASE 280
  - placement 270
  - REP 449
  - RLD 448
  - summary 270
  - TXT 448
- linking MSHP phases 267
- linking phases 169
- linking PTFs 394
- LIOCS operand (AFFECTS) 420
- LIST command (librarian) 310
- LIST operand (MOVE command) 318
- LIST operand (SETDF command) 209
- LIST option 170, 226
- LIST statement 396
- LISTDIR command (librarian) 311
- listing
  - directories 311
  - I/O device assignments 141
  - members 310
  - search chains 130
  - sublibrary definitions 130
  - system directory list (SDL) 312
- listing service tape/file 396
- LISTIO command/statement 141
- LISTLOG 226
- LISTX option 170, 226
- LNKEDT (program name of linkage editor) 267
- load address 280
- load lists in private sublibraries 205
- load parameter, IPL 9
- loading
  - forms control buffers (FCB) 125, 356
  - print buffers 125, 145
  - universal character-set buffers (UCB) 145, 356
- local fix
  - installing 382
  - removing 419
- LOCAL operand 165
- LOCK command (librarian) 314
- lock file
  - defining 22
  - size of 23
- locked members
  - copying 303
  - defining 314
  - displaying (LISTDIR) 313
  - moving 318
  - restoring 328
  - unlocking 333
- locked resources, releasing 243
- locking library members 314
- LOG command/statement 143
- LOG option 143, 170, 226
- LOG option (IPL) 11
- logging job control 143
  - on SYSLOG 143
  - on SYSLST 143
  - OPTION statement 170
  - source statements 170
  - STDOPT statement 226
  - suppressing 158, 170
- logical operator 118
- logical transient area (LTA) 205
- logical unit (EXTENT statement) 106
- logical unit assignments 365
- logical units
  - assigning 60
  - closing 82
  - defining number of 159
  - ignoring 63, 83
  - unassigning 63, 83
- logical units needed 365
- LOGSRC option 54, 170
- LOOKUP statement 397
- lowercase letters 372
- LTA (logical transient area) 205
- LUCB command 145

## M

- macro 373
  - assembling 417
  - regenerating 431, 433
- macro entry, removing from history file 407
- MACRO statement 296
- MACRO=member-name operand
  - GENERATE 431
  - INFLUENCES 433
  - LOOKUP 400
  - REMOVE 407

MACRO=member-name operand (*continued*)  
 TAILOR 417

macros  
 de-editing 345  
 renumbering 353  
 sequence-numbering 349  
 updating 348, 350, 352  
 verifying 354

MACROS=member-name operand  
 AFFECTS 421  
 COMPRISES 424

magnetic tape control 155

Maintain System History Program (MSHP)  
 overview 2

mandatory specifications 371

MAP command 146

MAP operand 273

mapping  
 all dynamic classes 147  
 dynamic class 148  
 partition 148  
 real storage 147  
 SVA 149  
 virtual storage 146

master catalog (VSE/VSAM), defining 16

member list, hexadecimal 310

member types 2, 286

member-name specification 373

member-type specification 373

members  
 cataloging 295  
 deleting 307  
 listing 310  
 locked 303, 318, 328  
 locking 314  
 punching 321  
 renaming 324  
 restoring 325  
 searching for 330  
 sequence numbering 334, 336  
 unlocking 333  
 updating 334, 336

MEMBERS operand (RETRACE) 413

MEND statement 296

MERGE HISTORY statement 401

merging history files 401

merging sublibraries 290, 301, 315

message suppression, IPL 9

migrating library members 321

minimum abbreviations of keywords 371

MNTC (mount complete) 142

MNTP (mount pending) 142

mode of operation, querying/altering 165

mode operand 66

mode setting for tapes 66

MODE statement (linkage editor) 278

MODIFY operand 209, 216

module 373  
 altering 422  
 generating 417  
 regenerating 431, 433

module entry, removing from history file 407

MODULE=member-name operand  
 GENERATE 431  
 INFLUENCES 433  
 LOOKUP 399  
 REMOVE 407

MODULE=member-name operand (*continued*)  
 TAILOR 417

modules 269

MODULES=member-name operand  
 AFFECTS 420  
 COMPRISES 424

MOVE command (librarian) 315

MRCZERO operand 204

MSECS command 153

MSG command 154

MSHP (EXEC LIBR) 289

MSHP (Maintain System History Program) 365  
 control statement operands 372  
 control statements 371  
 default values 371  
 history files 368  
 internal work file 368  
 overview 2  
 program name (MSHP) 365  
 reappearing the history file 370  
 restrictions 365, 369  
 return codes 370  
 work files 365, 368

MSHP (program name for the Maintain System History Program) 365

MSHP control statement operands  
 coding conventions 372  
 rules for writing frequently used 372  
 syntax rules 372

MSHP control statements  
 coding conventions 371  
 rules for writing 371  
 syntax rules 371

MSHP phases, linking 267

MSHP return codes 370

MTC command/statement 155

multi-volume files 108

multiple levels of program 373

multiple-partition allocation 57

## N

nested procedures 53

nesting levels 53

new-text operand (ALTER) 422

NEWVOL command 157

NOAUTO operand 273, 282

NOCHK operand 70, 145, 356

NOCOVER operand (LIST) 397

NODOCUMENT operand (LIST) 397

NODUMP operand 81

NODUMP option 169

NOLOG command/statement 158

NOLOG operand (OPTION statement) 143

Non-Parallel Application (NPA) operand 102

non-parallel share, NPS 193

non-volatile storage (NVS) 73

NOPDS specification (supervisor parameters command) 12

NOSLISKIP option 172

NOSYSDUMP operand 81

NOT operand (REQUIRES) 440

not-ready status, simulating 160

notation, syntax 2

notations, command 2

NOXREF operand  
 EXECUTE 430  
 LIST 397

NPA (Non-Parallel Application) operand 102  
 NPARTS operand 35  
 NPGR command 159  
 NPS, non-parallel share 193  
 NULMSG operand 125, 145, 241, 356, 357  
 number-of-blocks operand 109  
 number-of-tracks operand 109  
 NVS (non-volatile storage) 73

## O

OBJ operand 126, 129, 130  
 object modules, including 276  
 OFFLINE command 160  
 offset operand (SCAN) 443  
 old-format statements  
   APPLY 375  
   CORRECT statement 384  
 old-text operand (ALTER) 422  
 omitting values 371  
 ON command (librarian) 319  
 ON command/statement 161  
 ON conditions, default 162  
 ON statement example 262  
 ON-conditions (librarian), overriding 320  
 ONLINE command 164  
 operands (librarian commands)  
   keyword 288  
   positional 288  
 operands of control statements and commands  
   \$ABEND 161  
   \$CANCEL 161  
   \$MRC 117  
   \$RC 117, 161, 319  
   'file-id' 86, 236  
   accounting information 124  
   ALT 67, 83  
   band-id 355  
   band-ID 70  
   BLKSIZE 88  
   BLOCK 241  
   BSF 156  
   BSR 156  
   BUFND 88  
   BUFNI 88  
   BUFSP 88  
   BURST 208, 214  
   CANCEL 274  
   CAT 88  
   CATALOG 127, 129  
   CHARS 209, 215  
   CISIZE 89  
   CMD 165  
   COLUMN 335  
   command-line (EXEC LIBR) 289  
   comparator 117, 161  
   CONN 165  
   CONTINUE 162, 319  
   COPIES 215  
   DA 87  
   date 87, 236  
   DATE 302, 317  
   DCHK 215  
   DEBUG 215  
   device-class 63, 84  
   DFLT 215  
   DISC 165

operands of control statements and commands (*continued*)  
 DISP 89, 238  
 DSE 156  
 DSF 88  
 DU 87  
 DUMP 81  
 EOD 296  
 EOF 322  
 ERG 156  
 ERRLMT 274  
 extent type 107  
 FBA 62  
 FCB 209, 216  
 file sequence number 108  
 file-id 86, 236  
 file-section-number 238  
 file-sequence-number 238  
 file-serial-number 238  
 filename 86, 236  
 FLASH 209, 216  
 FOLD 70, 145, 241, 355, 356  
 FORCE 81, 161  
 FORMAT=HEX 310  
 FORMAT=IEBUdpdte 322  
 FORMAT=NOHEADER 322  
 FORMAT=OLD 321  
 FORMS 125, 209, 216  
 FSF 156  
 FSR 156  
 generation-number 238  
 generic (librarian) 288  
 GO 101  
 GOTO 162, 319  
 HC 204  
 ID 328  
 IGN 63, 83  
 INIT 216  
 ISC 87  
 ISE 87  
 jobname 124  
 label 113, 308  
 LINECT 204  
 LIST 209, 318  
 LOCAL 165  
 logical unit (EXTENT statement) 106  
 MAP 273  
 member type 286  
 mode 66  
 MODIFY 209, 216  
 MRCZERO 204  
 MSHP (EXEC LIBR) 289  
 NOAUTO 273, 282  
 NOCHK 70, 145, 356  
 NODUMP 81  
 NOLOG 143, 158  
 NOSYSDUMP 81  
 NPA 102  
 NULMSG 125, 145, 241, 356, 357  
 number-of-blocks 109  
 number-of-tracks 109  
 OBJ 126, 129, 130  
 operator 118  
 origin 280  
 OUTPUT 313  
 PARM 101  
 PARM=command line (EXEC LIBR) 289  
 PARM=MSHP (EXEC LIBR) 289

operands of control statements and commands (*continued*)

- PARTDUMP 81
- password 116
- PBDY 282
- PERM 67
- PHASE 126, 129, 130
- PHOLD 184
- PRELEASE 184
- PROC 102, 127, 129, 130
- RCLST 204
- RCPCH 204
- RCZERO 205
- REAL 99
- RECORDS 89
- RECSIZE 90
- relative-block 108
- relative-track 108
- REMOTE 165
- REPLACE 296, 302, 306, 317, 328
- RESET 209
- REUSE 297, 306
- REW 156
- RF 205
- RUN 156
- SAVE 334
- SCAN 327
- SD 87
- SDL 205, 312
- SEARCH 127, 129
- SELFACT 165
- SEP 217
- SEQUENCE 334
- sequence number 108
- serial number (EXTENT statement) 107
- set identifier 238
- SHR 68
- SIZE 99
- SMAP 274
- SORT 312
- SOURCE 126, 129, 130
- space-id 58
- split-cylinder-track 109
- SVA 282
- SVAPFIX 282
- sys-id 243
- SYSDUMP 81
- TEMP 67
- TRAIN 145
- TRC 217
- UA 63
- UNIT 310, 313
- UPSI 207
- user-id 116
- version-number 238
- VOL 68
- volume-sequence-number 238
- VSAM 87
- WTM 156
- OPERATE command 165
- operation code 365, 371
- operation mode 165
- operation, mode of 165
- operator communications routine 154
- OPTION statement 167
- optional values 371
- options, standard job control 225
- OR statement 436

- origin operand 280
- OSA Express Adapter 13
- OUTPUT operand (librarian) 313
- overriding
  - system date 85
  - system defaults 167
- overriding ON conditions 320
- overview
  - attention routine commands 47
  - job control statements and commands 47
- overview of control statements 366
  - detail control statements 367
  - function control statements 366

## P

- padding of volume serial number 4
- page data set
  - defining 24
  - multi-extent 25
  - system without 12
- page fixing, setting limits 212
- page-boundary alignment 282
- paper forms (3800 printer) 208
- parameterized procedures 51
- PARM operand 101
- PARM=command-line (EXEC LIBR) 289
- PARM=MSHP (EXEC LIBR) 289
- PARSTD option 171
- PARTDUMP operand 81
- PARTDUMP option 169
- partition
  - allocating storage 56
  - assignments, listing of 141
  - balancing 153, 179
  - changing priorities 182
  - deactivating 242
  - defining 56
  - deleting 58
  - device assignments, listing of 141
  - GETVIS 100
  - GETVIS area, displaying 111
  - GETVIS size, altering 219
  - I/O device assignments, listing of 141
  - label group 171
  - labels, adding 171
  - labels, deleting 171
  - librarian size 289
  - listing I/O assignments 141
  - logical units, number of 159
  - maximum number of 35
  - priority 178
  - priority for I/O requests 182
  - restarting 221
  - setting priorities 182
  - shared 37
  - size 99
  - starting 71, 221
  - stopping 228, 242
  - unbatching 115
- partition balancing 179
  - PRTY SHARE command 181
- partition-related procedure 102
- PASIZE definition 36
- passing parameters 101, 103
- passing POWER commands 184
- passing symbolic parameters 103



password operand 116  
password, defining 116  
PATCH statement 403  
patching a phase 403  
PAUSE command/statement 176  
PBDY operand 282  
PC (ESD type) 450  
PERM operand 67  
permanent assignment 60, 67  
PERSONALIZE statement 403  
personalizing a history file 403  
PFI area (above 16MB) 150  
PFI area (below 16MB) 150  
PFIing pages, setting limits 212  
phase 373  
    altering 422  
    generating 417  
    regenerating 431, 433  
phase entry, removing from history file 407  
PHASE operand 126, 129, 130  
PHASE statement (linkage editor) 280  
PHASE=member-name operand  
    GENERATE 431  
    INFLUENCES 433  
    LOOKUP 399  
    REMOVE 407  
    TAILOR 417  
phases  
    addressing mode 267, 278  
    alignment 282  
    cataloging 168, 280  
    chaining 128  
    linking of 169  
    load address 280  
    loading 205  
    names 280  
    names (FCB, UCB) 355  
    residence mode 267, 278  
    SVA-eligible 282  
PHASES=member-name operand  
    AFFECTS 420  
    COMPRISES 424  
PHOLD operand 184  
PHONE operand (PERSONALIZE) 405  
plus (+) sign 372  
positional operands 288  
POWER commands 184  
powering off the CPU 185  
PR (ESD type) 450  
PRE operand (REQUIRES) 439  
PRELEASE operand 184  
prerequisite 373  
    condition 439  
    negative 436  
    PTFs 394, 439  
preventive service, applying 396  
print band (4248) 70, 355  
print buffer loading, automatic 359  
print buffers, loading of 125, 145  
print-band ID (4248) 355  
printing  
    cross-reference list (of PTFs, APARs) 397  
    history file 386  
    selective cover letters 437  
    service tape/file 396  
printing job control statements 54  
PRINTLOG 226  
priority of I/O requests 182  
priority of partitions 178, 182  
private area, definition of 36  
private code 450  
PROC command/statement 177  
PROC operand 102, 127, 129, 130  
PROC statement example 263  
PROC statement in parameterized procedure 51  
procedures  
    calling 98  
    cataloged 177  
    coding 177  
    nested 53  
    nesting levels 53  
    parameterized 51  
    partition-related 102  
product 373  
    archiving 376  
    backing up 379  
    base 389  
    compatible 389, 422  
    copying to tape 379  
    listing (RETRACE) 412  
    removing from history file 406  
    restoring to disk 409  
    superseded 389  
product code 373  
product operand  
    ARCHIVE 377  
    REMOVE 406  
PRODUCT operand 373  
    INSTALL PRODUCT 390  
    RESTORE PRODUCT 409  
product specification 373  
PRODUCT=product operand  
    BACKUP PRODUCT 381  
    EXCLUDE 429  
    INCLUDE 432  
    LOOKUP 398  
    RESIDENCE 408  
PRODUCTION INTO operand  
    INSTALL PRODUCT/SYSRES 391  
    RESTORE PRODUCT/SYSRES 410  
PRODUCTION operand (BACKUP PRODUCT) 381  
production part (of shipment package) 391  
production sublibrary 380, 381  
PRODUCTION=lib.sublib operand  
    RESIDENCE 408  
PRODUCTS operand (RETRACE) 412  
program execution 98  
program number (of component) 372  
program switches, setting 207  
programmer logical units, defining number of 159  
PROGRAMMER operand (PERSONALIZE) 405  
programs  
    ESERV 345  
    librarian (LIBR) 289  
    linkage editor (LNKEDT) 267  
    Maintain System History File (MSHP) 365  
    system buffer load program (SYSBUFLD) 355  
PRTY command 178  
PRTY SHARE command 181  
PRTYIO command 182  
pseudo-register 450  
PTF 373  
    application 374  
    archiving 376

PTF (*continued*)  
 backout 375, 392, 394  
 cover letter 397  
 including in service 432  
 indirect application 375  
 installation 392  
 linking 394  
 listing (RETRACE) 413  
 prerequisite 394  
 reinstalling 394  
 removing 407  
 revoking 375, 414  
 superseded 444  
 PTF statement 436  
 ptf-number operand  
 APPLY 375  
 REVOKE 414  
 SUPERSEDES 444  
 PTF=ptf-number operand  
 ARCHIVE 378  
 EXCLUDE 429  
 INCLUDE 432  
 LOOKUP 399  
 REMOVE 407  
 PTFS operand (RETRACE) 413  
 PUB entries 13  
 PUNCH command (librarian) 321  
 punch output format 321  
 PUNCH statement (ESERV) 347  
 punching members 321  
 purpose of MSHP statements 365  
 purpose of statements 366  
 PWR command/statement 184  
 PWROFF command 185

## Q

QUERY command/statement 186  
 QUERY SCSI 191  
 QUERY TD display, interpreting the data 192  
 querying  
 data spaces 186  
 standard options setting 186  
 querying mode of operation 165  
 question marks (??) 372

## R

R-SIZE, mapping 151  
 RC command 194  
 RCLST operand 204  
 RCPCH operand 204  
 RCZERO operand 205  
 re-assigning devices 157  
 ready status, simulating 164  
 REAL mode 99  
 real storage definition 36  
 record size specification 89  
 recorder file  
 creating 205  
 defining 16  
 recording system statistics 202  
 recreating system status 375  
 RECSIZE operand 90  
 redirecting  
 librarian input 309

redirecting (*continued*)  
 librarian output 310, 313  
 REDISPLAY command 195  
 regenerating phases, modules, macros 431  
 reinstalling PTFs 394  
 relative block 108  
 relative track, calculation of 108  
 relative-block operand 108  
 relative-track operand 108  
 RELEASE command (librarian) 323  
 release number 373, 375, 388  
 RELEASE operand (INCORPORATE) 388  
 releasing library space 323  
 releasing locked resources 243  
 relocation dictionary, printing 226  
 relocation list dictionary (RLD) 269  
 REMOTE operand 165  
 REMOVE statement 406  
 removing entries from history file 406  
 RENAME command (librarian) 324  
 renaming sublibraries or members 324  
 renumbering macros 353  
 REP statement 449  
 repairing the history file 370  
 REPLACE operand 296, 302, 306, 317, 328  
 REPLACE statement 438  
 replacing member lines 340  
 REPLID command 199  
 reply-ID, displaying 199  
 req-list operand (REQUIRES) 440  
 requesting communication 194  
 REQUIRES statement 439  
 RESERV command 200  
 reserved status, resetting 110  
 reserved sublibrary 375, 395  
 reserving device for VSE/VSAM space 200  
 RESET command/statement 201  
 RESET operand 209  
 resetting  
 assignments 201  
 I/O assignments 201  
 LIBDEFs 201  
 library definitions 201  
 reserved status 110  
 tape mode 201  
 residence mode  
 assigning via MODE statement 278  
 residence mode (RMODE) 267  
 RESIDENCE statement 408  
 RESOLVES statement 441  
 resource unlocking 243  
 RESTART operand (INSTALL SERVICE/BACKOUT) 394  
 RESTART statement 442  
 restart-line operand (RESTART) 442  
 restarting a job 394  
 restarting checkpointed programs 203  
 restarting correction of edited macros 442  
 restarting partitions 221  
 RESTORE command (librarian) 325  
 RESTORE HISTORY statement 411  
 RESTORE PRODUCT/SYSRES statement 409  
 restoring a history file 411  
 restoring shipment tape to disk 409  
 restrictions 365, 369  
 retention period 87, 236  
 RETRACE statement 412

- return codes
  - conditional job control 50
  - librarian 285
  - linkage editor 269
  - MSHP 370
  - testing 50, 117, 261, 319
- REUSE attribute 301
  - changing 297
  - overriding 323
- REUSE operand 297, 306
- REVOKABLE operand
  - APPLY 375
  - CORRECT 384
  - INSTALL SERVICE 394
- REVOKE statement 414
- revoking a PTF 375, 414
- REW operand 156
- rewinding tape 156
- rewinding/unloading tape 156
- RF operand 205
- RLD (relocation list dictionary) 269
- RLD option 171, 226
- RLD statement 448
- RMODE (residence mode)
  - assigning via MODE statement 278
- ROD command 202
- RSIZE definition 36
- RSTRT statement 203
- rules
  - for coding frequently used operands 372
  - for writing MSHP control statements 371
- RUN operand 156

## S

- SADUMP option 171, 226
- SAVE operand 334
- SCAN operand 327
- SCAN statement 443
- SCANCEL option 172, 227
- scanning a phase string 443
- scope of symbolic parameter 53
- SCSI definitions 232
- SD (ESD type) 450
- SD operand 87
- SD operand (INSTALL SERVICE) 395
- SDL (system directory list) 205
  - listing 312
- SDL operand 205
- SDL operand (librarian) 312
- search chains
  - defining 126
  - dropping 129
  - listing 130
  - types of 126
- SEARCH command (librarian) 330
- SEARCH operand 127, 129
- searching for a library member 330
- searching for a phase string 443
- second history file 368
- section definition 450
- secured file 88
- security checking, activating 34
- SELECT statement 415
- SELFACT operand 165
- SEP operand 217
- SEPARATE operand (LIST) 397

- separators
  - blank 372
  - comma 372
  - comment 372
- sequence numbering 334
  - example 295
- sequence of detail control statements 371
- SEQUENCE operand 334
- sequence-numbering macros 349
- serial number 107
- service dialog (VSE/ESA) 375, 395
- service file 392, 394
  - listing 396
  - printing 396
- SERVICE operand (INSTALL SERVICE) 394
- service tape 392
  - documentation, printing 396
  - listing 396
  - printing 396
- service, applying 396
- SERVICETAPE operand (LIST) 396
- SET command (IPL) 26
- SET command (job control) 204
- SET XPCC command (IPL) 28
- SET ZONEBDY command 30
- SET ZONEDEF command 29
- set-identifier operand 238
- SETDF command 208
- SETPARM command/statement 210
- SETPARM in parameterized procedure 51
- SETPARM statement example 263
- SETPFIX command/statement 212
- SETPRT command/statement 214
- SETPRT example 218
- setting
  - line count 204
  - options (job control) 225
  - symbolic parameters 210
  - system date 26
  - system values 204
  - time zone 26, 251
  - time-of-day (TOD) clock 26
- shared partitions
  - reserving storage for 37
- shared virtual area (SVA) 205
  - defining 32
- sharing disk devices 14
- sharing VM DB2 data bases 28
- SHR operand 68
- simulating
  - device-not-ready status 160
  - device-ready status 164
- single quotes (') 372
- single-partition allocation 56
- SIZE command 219
- size of extent 109
- SIZE operand 99
- skipping statements 113
- slashes (//) 48
- SLISKIP option 172
- SMAP operand 274
- SOFTREJECT operand (ARCHIVE) 378
- SORT operand (librarian) 312
- SOURCE operand 126, 129, 130
- space-id operand 58
- special signs 372
- specifying supervisor parameters 11

- split cylinder 109
- split-cylinder-track operand 109
- SPLIT=split-track operand (DEFINE HISTORY) 427
- SPSIZE operand 37
- stand-alone dump 171
- standard
  - buffer load phases 357
  - FCB/UCB image phases 357
  - job control options 225
  - label group 171
  - labels 172
  - labels, adding 173
  - labels, deleting 173
- standard tape labels 379, 381, 390
- START command 221
- starting partitions 221
- statement notation explained 2
- statement overview 366
- static partition
  - allocating storage 56
  - altering storage 59
  - displaying storage 91
  - dumping 93
  - GETVIS area, displaying 111
  - I/O request priority setting 182
  - logging job control statements 144
  - priority setting 178
- STATUS command 222
- status of devices, displaying 222
- status of tasks, displaying 222
- STDLABEL option 172
- STDOPT command/statement 225
- STOP command 228
- stopping partitions 228
- storage allocation rules 38
- storage map 146
- sublib specification 373
- SUBLIB=AE option 173
- SUBLIB=DF option 173
- sublibraries
  - accessing 290
  - defining 126, 305
  - defining number of 34
  - definitions, holding 115
  - definitions, listing of 130
  - deleting 307
  - in use 285
  - merging 290, 301, 315
  - renaming 324
  - restoring 325
- sublibrary
  - member 373
  - name 373
  - production/generation 380
- SUBLIBRARY= operand (PATCH) 403
- superseded
  - products 389
  - PTFs 444
- SUPERSEDES statement 444
- supervisor buffers, defining 34
- supervisor name (IPL) 11
- supervisor parameters command (IPL) 11
- supervisor parameters, specifying 11
- suppressing job control logging 158
- suspending processing 176
- SVA (shared virtual area) 205
  - defining 32
- SVA command (IPL) 32
- SVA operand 282
- SVA-eligible phases 282
- SVA, mapping 149
- SVAPFIX operand 282
- SXREF option 174, 227
- SYM option 173, 227
- symbolic parameters
  - at level n 210
  - at POWER job level 210
  - at system level 210
  - concatenation 52
  - defaults 177
  - description 51
  - example 52
  - format of 51
  - initializing 177
  - passing 101, 103
  - scope 53
  - setting 210
  - testing 117
- syntax notation explained 2
- syntax of commands 2
- syntax rules 371, 372
- syntax rules (job control) 48
- syntax symbols 2
- SYS command (IPL) 34
- sys-id operand 243
- SYSBUFLD (program name for system buffer load program) 355
- SYSBUFLD (system buffer load) control statements
  - BANDID 355
  - FCB 356
  - overview 355
  - UCB 356
- SYSBUFLD (system buffer load) program 355
  - overview 2
- SYSCAT, defining 16
- SYSDEF command/statement 229
- SYSDEF SCSI command/statement 232
- SYSDEF TD,RESETCNT command 234
- SYSDEF TD,START command/statement 234
- SYSDEF TD,STOP command 234
- SYSDUMP operand 81
- SYSDUMP option 173, 227
- SYSDUMPC option 173, 227
- SYSECHO command 235
- SYSIN format 388
- SYSIN input device 365
- SYSIPT, assigning 61
- SYSLIB 391
- SYSLOG
  - defining 7
  - logging on 143
- SYSLOG input device 365
- SYSLOG, displaying on 397
- SYSLST
  - assigning 61, 62
  - extent full 204
  - logging on 143
  - page length 226
- SYSOUT, assigning 62
- SYS Parm option 174
- SYS PCH
  - assigning 62
  - extent full 204
- SYSRDR, assigning 61

- SYSREC
  - assigning 61
  - defining 16
  - writing to 202
- SYSRES operand
  - INSTALL SYSRES 390
  - RESTORE SYSRES) 409
- SYSRES package, installing 389
- system
  - date, overriding 85
  - date, setting 26
  - defaults, overriding 167
  - label information 172
  - phases, defining 128
  - statistics, recording 202
  - sublibraries, chaining 127
  - sublibrary, access rights 127
- system buffer load (SYSBUFLD) program 355
  - overview 2
  - program name (SYSBUFLD) 355
- system directory list (SDL) 205
  - listing 312
- system GETVIS area
  - displaying 111
- system history file 368, 373
  - backing up 294, 378
  - copying 381
  - copying to tape 378
  - creating 385
  - defining 16, 426
  - displaying 397
  - dumping 386
  - identifying 403
  - initializing 385
  - making entries in the 376
  - personalizing 403
  - printing information from 412
  - removing entries from 406
  - usage 369
- SYSTEM operand
  - BACKUP HISTORY 379
  - COPY HISTORY 382
  - CREATE HISTORY 386
  - DEFINE HISTORY 426
  - DUMP HISTORY 387
  - MERGE HISTORY 402
  - RESTORE HISTORY 411
- system programs invoked by MSHP
  - assembler 365, 368
  - librarian 365, 368
  - linkage editor 365, 368
- system recorder file
  - creating 205
  - defining 16
- system sublibrary 375
- system-ID 243

## T

- TAILOR statement 416
- tape 373
  - assignment (alternate) 67
  - backout 392, 394
  - control 155
  - copying history file to 378
  - copying product to 379
  - file creation date 236

- tape (*continued*)
  - file names 236
  - file-id 236
  - installing from 390
  - labels 236, 379, 381, 390
  - standard labels 379, 381, 390
- tape mark, writing 156
- tape mode
  - resetting 201
  - setting 66
- TAPELABEL=filename operand
  - BACKUP HISTORY 379
  - BACKUP PRODUCT 381
  - INSTALL PRODUCT/SYSRES 390
  - RESTORE HISTORY 411
  - RESTORE PRODUCT/SYSRES 409
- tapeless system 390
- TAPES= operand (INSTALL SERVICE) 394
- target libraries for INSTALL PRODUCT/SYSRES 391
- task status, displaying 222
- telecommunication balancing 240
- teleprocessing balancing 178, 240
- TEMP operand 67
- temporary assignment 60, 67
- TERM option 174, 227
- TEST command (librarian) 332
- testing
  - parameters 117
  - return codes 50, 117, 261, 319
- text record 269
- time slice 153
- time zone, setting 26, 251
- time-of-day (TOD) clock 26
- timer setting 26
- TLBL statement 236, 390
- to-line operand (DELETE) 428
- to-line operand (REPLACE) 438
- TOD (time-of-day) clock 26
- TP balancing 240
- TPBAL command 240
- tracks per cylinder 108
- train image 361
- TRAIN operand 145
- TRC operand 217
- TXT statement 448
- type (of sublibrary member) 373
- TYPE=member-type operand
  - AFFECTS 421
  - COMPRISES 424
  - GENERATE 431
  - INFLUENCES 433
  - LOOKUP 400
  - REMOVE 407
  - TAILOR 417
- types of search chains 126

## U

- UA operand 63
- UCB (universal character-set buffer)
  - loading via SYSBUFLD 356
  - loading via UCS command 241
- UCB image phases
  - creating 359
  - description 357
  - format 361
  - names 355

- UCB image phases (*continued*)
  - non-standard 357
  - standard 357
- UCB statement (SYSBUFLD) 356
- UCS (universal character set), loading 241
- UCS buffer loading 241
- UCS command 241
- unassigning logical units 63, 83
- UNBATCH command 242
- underlined option 371
- underlining 371
- UNDO statement 384, 419
- UNIT operand 310, 313
- UNIT=SYSxxx operand (DEFINE HISTORY) 427
- universal character set (UCS), loading 241
- universal character-set buffer (UCB)
  - loading via SYSBUFLD 356
  - loading via UCS command 241
- unloading tape 156
- UNLOCK command 243
- UNLOCK command (librarian) 333
- unlocking library members 333
- unlocking resources 243
- unlocking the console 176
- UPDATE command (librarian) 334
- UPDATE subcommands (librarian)
  - )ADD 337
  - )DEL 338
  - )END 339
  - )REP 340
  - overview 336
- updating macros 348, 350, 352
- updating members 334
- uppercase letters 372
- UPSI (user program switch indicator) 207, 244
- UPSI operand 207
- UPSI statement 244
- usage of auxiliary history file 369
- user identification 116
- user program switch indicator (UPSI) 207, 244
- user-id operand 116
- user-id, defining 116
- USRLABEL option 174

## V

- V-SIZE, mapping 152
- VCTE (Volume-Characteristic-Table-Entries) 164
- VDISK command/statement 245
- verification message 362
  - suppressing 241, 356, 357
- VERIFY statement 445
- verify-line operand (VERIFY) 445
- verifying a correction 445
- verifying macros 354
- version-number operand 238
- VIO (virtual I/O area) specification (IPL) 12
- virtual disk
  - adding 14
  - assigning 64
  - defining layout (VDISK) 245
  - initializing (VDISK) 245
- virtual I/O (VIO) area, size of 12
- virtual storage
  - allocating 56
  - altering 59
  - displaying 91

- virtual storage (*continued*)
  - dumping 92
  - map of 146
  - program size 219
  - system size 12
- virtual tape
  - defining VSAM data set (VTAPE) 249
  - releasing the access (VTAPE) 249
- VM DB2 data base 28
- VM minidisk 382, 386
- VOL operand 68
- volume assignment, altering 157
- VOLUME command 247
  - output example 248
- volume serial number 107
  - padding 4
- Volume-Characteristic-Table-Entries (VCTE) 164
- volume-sequence-number operand 238
- VPOOL specification (IPL) 11
- VSAM operand 87
- VSAM-managed space (for work files) 369
- VSE system control programs called by MSHP 368
- VSE/ESA service dialog 395
- VSE/POWER commands 184
- VSE/VSAM 389
  - buffer space 88
  - catalog 88
  - data buffers 88
  - file disposition 89
  - file extents 107
  - index buffers 88
  - job catalog 88
  - master catalog 88
  - master catalog, defining 16
  - private user catalog 88
  - reserving device for 200
  - space management for SAM 90
- VSIZE specification (IPL) 12
- VTAM gateway information 28
- VTAPE command/statement 249

## W

- weak external reference 450
- WITH=product operand (COMPATIBLE) 423
- work files used by MSHP 365, 368
- writing tape mark 156
- WTM operand 156
- WX (ESD type) 450

## X

- XPCC/APPC/VM support 28
- XREF operand (EXECUTE) 430
- XREF operand (LIST) 397
- XREF option 174, 227

## Y

- Year 2000 support
  - DLBL statement 87
  - End-of-Job Statement 255
  - Job statement 124
  - linkage editor 268
  - TLBL statement 237

## **Z**

ZONE statement 251





---

# Readers' Comments — We'd Like to Hear from You

IBM z/VSE  
System Control Statements  
Version 3 Release 1

Publication No. SC33-8225-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you?  Yes  No

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

\_\_\_\_\_  
Name

\_\_\_\_\_  
Address

\_\_\_\_\_  
Company or Organization

\_\_\_\_\_  
Phone No.



Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Entwicklung GmbH  
Information Development  
Department 3248  
Schoenaicher Strasse 220  
71032 Boeblingen  
Germany



Fold and Tape

Please do not staple

Fold and Tape





File Number: S370/S390-34  
Program Number: 5609-ZVS

Printed in USA

SC33-8225-00



Spine information:



IBM z/VSE

z/VSE System Control Statements

Version 3 Release 1

SC33-8225-00