

System Modification Program Extended

SC28-1107-13

**Reference**

Release 8.1







# System Modification Program Extended

SC28-1107-13

## Reference

Release 8.1

---

**Note!**

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

| **Fourteenth Edition (September 1994)**

- | This book replaces the previous edition, SC28-1107-12, which is now obsolete. Changes or additions to text and illustrations are indicated by a vertical line to the left of the change.
- | This edition applies to System Modification Program Extended (SMP/E) Release 8.1 and to all subsequent releases and modifications unless otherwise indicated in new editions.

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

IBM welcomes your comments. A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments to:

| IBM Corporation, Department 52QA, Mail Station 911  
Enterprise Drive  
Kingston, NY 12401-1099  
United States of America

FAX: Your International Access Code +1+914+385-0662

IBMLink (United States customers only): KGNRCF at KGNVMC  
IBM Mail Exchange: USIB27BQ at IBMMAIL  
Internet: kgnrcf@vnet.ibm.com

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

© Copyright International Business Machines Corporation 1982, 1994. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.



---

# Contents

<b>Notices</b> .....	xxi
Programming Interface Information .....	xxi
Trademarks .....	xxii
<b>About This Book</b> .....	xxiii
What This Publication Contains .....	xxiii
<b>Summary of Changes</b> .....	xxv
SMP/E Release 8.1 (September 1994) .....	xxv
Changes for SC28-1107-13 .....	xxvi
SMP/E Release 8 (March 1994) .....	xxvi
<b>Chapter 1. Syntax Notation and Rules</b> .....	1
How to Read the Syntax Diagrams .....	1
Syntax Rules .....	2
<b>Chapter 2. The ACCEPT Command</b> .....	5
Zones for SET BOUNDARY .....	5
Syntax .....	6
Operands .....	7
Syntax Notes .....	17
Data Sets Used .....	18
Usage Notes .....	19
Adding New Elements to the Distribution Libraries .....	19
DISTLIB Operand Checking .....	19
DISTSRC, ASSEM, and DISTMOD Operands .....	20
Alias Processing .....	20
ACCEPT CHECK Facility .....	21
SYSMOD Termination .....	21
ACCEPT Termination .....	23
Automatic Reinstallation of SYSMODs .....	23
Output .....	24
Examples .....	24
Example 1: Accepting All SYSMODs from a Given Source .....	24
Example 2: Accepting All SYSMODs for Selected Functions .....	25
Example 3: Accepting with the GROUP Operand .....	25
Example 4: Accepting with the GROUPEXTEND Operand .....	26
Example 5: Accepting with the CHECK Operand .....	26
Example 6: Combining ACCEPT Operands .....	27
Example 7: Doing ACCEPT before APPLY .....	27
Example 8: Installing Service for All ESO Service Levels .....	28
Example 9: Excluding SYSMODs with Certain Source IDs .....	28
Processing .....	28
SYSMOD Selection .....	28
SYSMOD Installation .....	34
Element Selection .....	38
Element Installation .....	41
Recording After Completion .....	47
Zone and Data Set Sharing Considerations .....	50

<b>Chapter 3. The APPLY Command</b>	53
Zones for SET BOUNDARY	53
Syntax	54
Operands	55
Syntax Notes	65
Data Sets Used	67
Usage Notes	67
Adding New Elements Other Than Modules to the Target Libraries	67
Adding New Modules to the Target Libraries	68
Checking the DISTLIB Operand	68
DISTSRC, ASSEM, and DISTMOD Operands	69
Use of the SMPMTS and SMPSTS as Target Libraries	70
Use of the SMPLTS Library	70
Alias Processing	70
APPLY CHECK Facility	71
SYSMOD Termination	71
APPLY Termination	72
Automatic Reapplication of SYSMODs	73
Output	73
Examples	73
Example 1: Applying All SYSMODs from a Given Source	74
Example 2: Applying All SYSMODs for Selected Functions	74
Example 3: Applying APARs and USERMODs	74
Example 4: Applying with the GROUP Operand	75
Example 5: Applying with GROUPEXTEND	75
Example 6: Applying with the CHECK Operand	76
Example 7: Combining APPLY Operands	76
Example 8: Installing Service for All ESO Service Levels	77
Example 9: Excluding SYSMODs with Certain Source IDs	77
Processing	77
SYSMOD Selection	77
SYSMOD Installation	82
Element Selection	87
Building Load Modules	90
Element Installation	91
Recording After Completion	102
Cross-Zone Processing	105
Global Zone SYSMOD Entries	106
Zone and Data Set Sharing Considerations	106
<b>Chapter 4. The CLEANUP Command</b>	107
Zones for SET BOUNDARY	107
Syntax	107
Operands	108
Data Sets Used	108
Output	108
Example: Using CLEANUP with the COMPRESS Operand	109
Processing	110
Zone and Data Set Sharing Considerations	111
<b>Chapter 5. The CONVERT Command</b>	113
Zones for SET BOUNDARY	114
Syntax	114

Converting CSI and SCDS Data Sets (SMP/E Release 4 or Earlier) . . . . .	114
Converting SMP4 Data Sets . . . . .	114
Operands . . . . .	115
Syntax Notes . . . . .	118
Data Sets Used . . . . .	119
Usage Notes . . . . .	119
Converting CSI Data Sets . . . . .	119
Converting SCDS Data Sets . . . . .	121
Reorganizing the CSI after Conversion . . . . .	122
Recovering After Conversion Fails . . . . .	123
Output . . . . .	123
Examples . . . . .	123
Example 1: Converting a Single-CSI System . . . . .	124
Example 2: Converting a Multiple-CSI System . . . . .	127
Example 3: Converting a Master CSI Data Set . . . . .	130
Example 4: Converting a CSI Data Set for Target or DLIB Zones . . . . .	131
Example 5: Converting an SCDS Data Set . . . . .	131
Example 6: Converting the SMP4 PTS . . . . .	133
Example 7: Converting the SMP4 CDS and CRQ . . . . .	134
Example 8: Converting the SMP4 ACDS and ACRQ by Entry Type . . . . .	135
Example 9: Recovering After Conversion Fails . . . . .	136
Processing . . . . .	137
CSI Processing . . . . .	137
SCDS Processing . . . . .	137
General Processing for Other Data Sets . . . . .	138
Special Processing for Other Data Sets . . . . .	139
Zone and Data Set Sharing Considerations . . . . .	141
<b>Chapter 6. The DEBUG Command . . . . .</b>	<b>143</b>
Zones for SET BOUNDARY . . . . .	143
Syntax . . . . .	143
Operands . . . . .	143
Syntax Notes . . . . .	145
Data Sets Used . . . . .	145
Output . . . . .	145
Examples . . . . .	145
Example 1: Tracing SMP/E Messages . . . . .	146
Example 2: Dumping Control Blocks and Storage Areas . . . . .	146
Example 3: Dumping a VSAM RPL Control Block . . . . .	147
Example 4: Dumping SMP/E Storage When Messages Are Issued . . . . .	147
Processing . . . . .	148
<b>Chapter 7. The GENERATE Command . . . . .</b>	<b>149</b>
Zones for SET BOUNDARY . . . . .	149
Syntax . . . . .	149
Operands . . . . .	150
Data Sets Used . . . . .	151
Usage Notes . . . . .	152
Output . . . . .	152
Examples . . . . .	152
Example 1: Using GENERATE to Install New Products . . . . .	152
Example 2: Reinstalling Products Not Included by SYSGEN . . . . .	153
Processing . . . . .	155

Target Zone Analysis .....	155
JCL Creation .....	160
Job Generation .....	162
Using the Output from GENERATE .....	164
Zone and Data Set Sharing Considerations .....	165
<b>Chapter 8. The JCLIN Command</b> .....	167
Zone for SET BOUNDARY .....	168
Syntax .....	168
Operands .....	168
Data Sets Used .....	170
Usage Notes .....	170
SYSMODs with Inline JCLIN .....	170
PARMLIB Members to Identify Opcodes in Assembler Text .....	171
Packaging JCLIN Input .....	171
Processing after System Generation or I/O Generation .....	172
Cross-Zone Relationships .....	172
Output .....	172
Examples .....	173
Example 1: JCLIN for Products with Special Utilities .....	173
Example 2: JCLIN for Products with Special Assembler Opcodes .....	173
Example 3: JCLIN for MOD Entries .....	174
Example 4: JCLIN for MAC and SRC Entries .....	177
Example 5: JCLIN for an Assembler Step to Create a SRC Entry .....	178
Example 6: JCLIN for Load Modules Residing in a Hierarchical File System .....	178
Processing .....	179
Summary .....	180
General JCLIN Coding Conventions .....	180
Processing Assembler Steps .....	183
Processing Copy Steps .....	185
Processing Link-Edit Steps .....	188
Processing Update Steps .....	198
Processing Other Utility Steps .....	198
Zone and Data Set Sharing Considerations .....	199
<b>Chapter 9. The LINK Command</b> .....	201
Zones for SET BOUNDARY .....	202
Syntax .....	202
Operands .....	202
Data Sets Used .....	203
Output .....	204
Example: Linking a GDDM Module into a CICS Load Module .....	204
Processing .....	205
Preparing for Linking .....	205
Linking the Load Modules .....	208
Zone and Data Set Sharing Considerations .....	209
<b>Chapter 10. The LIST Command</b> .....	211
Zones for SET BOUNDARY .....	211
Syntax .....	211
Distribution Zone and Target Zone Syntax .....	212
Global Zone Syntax .....	214
SMPLOG Syntax .....	215

SMPSCDS Syntax	215
Operands	215
Syntax Notes	229
Data Sets Used	230
Usage Notes	230
Output	230
Examples	231
Example 1: List All the Entries in a Particular Zone	231
Example 2: List All the Entries of a Particular Type	231
Example 3: List Specific Entries	231
Example 4: List Entries Applicable to Specific FMIDs	232
Example 5: Check Which SYSMODs Are Received But Not Installed	232
Example 6: Check Whether SYSMODs Are Installed in the Related Zone	232
Example 7: Compare the SYSMODs Installed in Two Zones of the Same Type	233
Processing	233
Mass-Mode Processing	234
Select-Mode Processing	234
Zone and Data Set Sharing Considerations	234
<b>Chapter 11. The LOG Command</b>	235
Zones for SET BOUNDARY	235
Syntax	235
Operands	235
Data Sets Used	236
Output	236
Examples	236
Example 1: Writing a Message	236
Example 2: Coding Parentheses Correctly	237
Example 3: Listing an SMPLOG Data Set	237
Processing	237
<b>Chapter 12. The RECEIVE Command</b>	239
Zones for SET BOUNDARY	239
Syntax	239
Operands	239
Syntax Notes	242
Data Sets Used	242
Usage Notes	243
Receiving SYSMODs Packaged in Relative Files	243
Defining an Installation-Wide Exit Routine for RECEIVE Processing	246
Output	246
Listings	246
Reports	247
Examples	247
Example 1: Receiving SYSMODs and HOLDDATA	247
Example 2: Receiving HOLDDATA Only	248
Example 3: Receiving SYSMODs Only	248
Example 4: Receiving Selected SYSMODs and HOLDDATA	248
Example 5: Receiving SYSMODs and HOLDDATA for a Specific FMID	248
Example 6: Receiving RELFILES from DASD	249
Processing	249
Processing Relative Files	249

Selecting SYSMODs	251
Selecting ++HOLD and ++RELEASE Statements	252
Processing SYSMODs	252
Processing ++ASSIGN Statements	253
Processing ++HOLD and ++RELEASE Statements	253
Zone and Data Set Sharing Considerations	254
<b>Chapter 13. The REJECT Command</b>	<b>255</b>
Zones for SET BOUNDARY	255
Syntax	255
Mass Mode Syntax	256
Select Mode Syntax	256
PURGE Mode Syntax	257
NOFMID Mode Syntax	257
Operands	257
Data Sets Used	262
Output	263
Reports	263
Statistics	263
Examples	264
Example 1: Rejecting All SYSMODs That Have Not Been Installed (Mass Mode)	264
Example 2: Rejecting All SYSMODs for a Specific Function (Mass Mode)	265
Example 3: Rejecting Selected SYSMODs That Have Been Applied (Select Mode)	265
Example 4: Rejecting Selected SYSMODs That Have Been Accepted and Applied (Select Mode)	265
Example 5: Rejecting HOLDDATA That Has No SYSMOD Entry (Select Mode)	265
Example 6: Rejecting SYSMODs That Have Been Accepted (PURGE Mode)	266
Example 7: Rejecting SYSMODs That Have Been Accepted and Applied (PURGE Mode)	266
Example 8: Rejecting SYSMODs for Undefined Functions (NOFMID Mode)	266
Example 9: Deleting Service for a Group of Source IDs	267
Example 10: Rejecting Selected SYSMODs That Have Been Superseded (Select Mode)	267
Processing	267
Selecting the Eligible SYSMODs and HOLDDATA	267
Processing the SYSMODs and HOLDDATA	271
Zone and Data Set Sharing Considerations	272
<b>Chapter 14. The REPORT CALLLIBS Command</b>	<b>275</b>
Zones for SET BOUNDARY	275
Syntax	275
Operands	275
Data Sets Used	276
Output	277
Reports	277
SMPPUNCH Output	277
Example: Using REPORT CALLLIBS	279
Processing	283
Zone and Data Set Sharing Considerations	286

<b>Chapter 15. The REPORT CROSSZONE Command</b> .....	287
Zones for SET BOUNDARY .....	287
Syntax .....	287
Operands .....	287
Data Sets Used .....	289
Usage Notes .....	289
Output .....	289
Reports .....	289
SMPPUNCH Output .....	290
Examples .....	291
Example 1: Using REPORT CROSSZONE .....	291
Example 2: Using REPORT CROSSZONE with Zones Controlled by Different Global Zones .....	294
Processing .....	295
Zone and Data Set Sharing Considerations .....	297
<b>Chapter 16. The REPORT ERRSYSMODS Command</b> .....	299
Zones for SET BOUNDARY .....	299
Syntax .....	299
Operands .....	299
Data Sets Used .....	301
Usage Notes .....	302
Output .....	302
Reports .....	302
SMPPUNCH Output .....	302
Example: Using REPORT ERRSYSMODS .....	304
Processing .....	305
Zone and Data Set Sharing Considerations .....	307
<b>Chapter 17. The REPORT SOURCEID Command</b> .....	309
Zones for SET BOUNDARY .....	309
Syntax .....	309
Operands .....	309
Data Sets Used .....	310
Output .....	310
Reports .....	310
SMPPUNCH Output .....	310
Examples .....	311
Example 1: REPORT SOURCEID (SYSMODIDS Operand Specified) .....	311
Example 2: REPORT SOURCEID (SYSMODIDS Operand Not Specified) .....	312
Example 3: REPORT SOURCEID (ZONES Operand Specified) .....	313
Processing .....	314
Zone and Data Set Sharing Considerations .....	315
<b>Chapter 18. The REPORT SYSMODS Command</b> .....	317
Zones for SET BOUNDARY .....	317
Syntax .....	317
Operands .....	317
Data Sets Used .....	318
Output .....	318
Reports .....	318
SMPPUNCH Output .....	318
Example: Using REPORT SYSMODS .....	321

Processing	325
Zone and Data Set Sharing Considerations	328
<b>Chapter 19. The RESETRC Command</b>	329
Zones for SET BOUNDARY	329
Syntax	329
Data Sets Used	329
Usage Notes	329
Examples	330
Example 1: Using RESETRC between Commands for One Zone	330
Example 2: Using RESETRC between Commands for Different Zones	330
Processing	331
<b>Chapter 20. The RESTORE Command</b>	333
Zones for SET BOUNDARY	333
Syntax	333
Operands	333
Data Sets Used	335
Usage Notes	336
Output	338
Examples	339
Example 1: Restoring a Single SYSMOD	339
Example 2: Restoring Multiple PTFs to Remove One PTF	340
Example 3: Restoring PTFs Using the GROUP Operand	340
Processing	341
SYSMOD Selection	341
Element Installation	342
Recording After Completion	346
Cross-Zone Processing	346
Global Zone SYSMOD Entries	347
Zone and Data Set Sharing Considerations	347
<b>Chapter 21. The SET Command</b>	349
Syntax	349
Operands	349
Data Sets Used	350
Usage Notes	350
Examples	350
Example 1: Receiving SYSMODs into the SMPPTS Data Set	351
Example 2: Applying SYSMODs to the Target Libraries	351
Example 3: Accepting SYSMODs to the Distribution Libraries	351
Example 4: Processing Multiple Commands in One Invocation of SMP/E	352
Example 5: Changing Which OPTIONS Entry Is Used	352
Example 6: Resolving Errors in Dynamic Allocation	352
Processing	353
Zone and Data Set Sharing Considerations	354
<b>Chapter 22. The UCLIN Command</b>	355
Zones for SET BOUNDARY	355
UCLIN and ENDUCL Syntax	355
Operands	357
UCL Statement Syntax	357
ASSEM Entry Syntax (Distribution and Target Zone)	359



BACKUP Entry Syntax (SMPSCDS Data Set)	359
Data Element Entry Syntax (Distribution and Target Zone)	360
DDDEF Entry Syntax (Distribution, Target, and Global Zone)	361
DLIB Entry Syntax (Distribution and Target Zone)	363
DLIBZONE Entry Syntax (Distribution Zone)	363
FMIDSET Entry Syntax (Global Zone)	364
GLOBALZONE Entry Syntax (Global Zone)	364
HFS Entry Syntax (Distribution and Target Zone)	365
LMOD Entry Syntax (Distribution and Target Zone)	366
MAC Entry Syntax (Distribution and Target Zone)	367
MOD Entry Syntax (Distribution and Target Zone)	368
MTSMAC Entry Syntax (SMPMTS Data Set)	369
OPTIONS Entry Syntax (Global Zone)	369
SRC Entry Syntax (Distribution and Target Zone)	370
STSSRC Entry Syntax (SMPSTS Data Set)	370
SYSMOD Entry Syntax (Distribution and Target Zone)	370
SYSMOD Entry Syntax (Global Zone)	374
TARGETZONE Entry Syntax (Target Zone)	374
UTILITY Entry Syntax (Global Zone)	374
ZONESET Entry Syntax (Global Zone)	375
Data Sets Used	375
Output	375
Usage Notes	375
Examples	376
Example 1: UCLIN to Change a Global Zone Entry	376
Example 2: UCLIN to Change a Target Zone Entry	376
Example 3: UCLIN to Change a Distribution Zone Entry	376
Processing	377
Zone and Data Set Sharing Considerations	378
<b>Chapter 23. The UNLOAD Command</b>	379
Zones for SET BOUNDARY	379
Syntax	379
Operands	380
Syntax Notes	387
Data Sets Used	388
Output	388
Examples	388
Processing	388
Mass-Mode Processing	389
Select-Mode Processing	389
Zone and Data Set Sharing Considerations	389
<b>Chapter 24. The ZONECOPY Command</b>	391
Zones for SET BOUNDARY	391
Syntax	391
Operands	391
Syntax Notes	392
Data Sets Used	392
Usage Notes	393
Output	394
Examples	394
Example 1: Copying a Target Zone to a Target Zone	394

Example 2: Copying a Distribution Zone to a Distribution Zone . . . . .	395
Example 3: Copying a Distribution Zone to a Target Zone . . . . .	395
Processing . . . . .	396
Zone and Data Set Sharing Considerations . . . . .	396
<b>Chapter 25. The ZONEDELETE Command . . . . .</b>	<b>399</b>
Zones for SET BOUNDARY . . . . .	399
Syntax . . . . .	399
Operands . . . . .	399
Syntax Notes . . . . .	400
Data Sets Used . . . . .	400
Usage Notes . . . . .	400
Output . . . . .	401
Examples . . . . .	401
Example 1: Deleting a Target Zone . . . . .	401
Example 2: Deleting a Distribution Zone . . . . .	402
Processing . . . . .	402
Zone and Data Set Sharing Considerations . . . . .	402
<b>Chapter 26. The ZONEEDIT Command . . . . .</b>	<b>405</b>
Zones for SET BOUNDARY . . . . .	405
Syntax . . . . .	405
Operands . . . . .	406
Syntax Notes . . . . .	408
Data Sets Used . . . . .	408
Output . . . . .	408
Examples . . . . .	409
Example 1: Editing DDDEF Entries . . . . .	409
Example 2: Conditionally Editing DDDEF Entries . . . . .	409
Example 3: Changing the SYSOUT Value . . . . .	409
Example 4: Changing the Zone Value in Cross-Zone Subentries . . . . .	410
Processing . . . . .	410
Zone and Data Set Sharing Considerations . . . . .	410
<b>Chapter 27. The ZONEEXPORT Command . . . . .</b>	<b>413</b>
Zones for SET BOUNDARY . . . . .	413
Syntax . . . . .	413
Operands . . . . .	413
Data Sets Used . . . . .	414
Usage Notes . . . . .	415
Output . . . . .	415
Example: Backing Up Target and Distribution Zones . . . . .	415
Processing . . . . .	415
Zone and Data Set Sharing Considerations . . . . .	416
<b>Chapter 28. The ZONEIMPORT Command . . . . .</b>	<b>417</b>
Zones for SET BOUNDARY . . . . .	417
Syntax . . . . .	417
Operands . . . . .	417
Data Sets Used . . . . .	419
Usage Notes . . . . .	419
Output . . . . .	419
Examples . . . . .	419

Example 1: Importing a Distribution Zone into a Target Zone	420
Example 2: Importing a Global Zone	420
Processing	420
Zone and Data Set Sharing Considerations	421
<b>Chapter 29. The ZONEMERGE Command</b>	<b>423</b>
Zones for SET BOUNDARY	423
Syntax	424
Operands	424
Data Sets Used	425
Usage Notes	425
Output	426
Examples	426
Example 1: Creating New Target Zone after System Generation	426
Example 2: Creating a Test Target System	428
Example 3: Creating a Test Distribution System	429
Processing	429
Zone and Data Set Sharing Considerations	431
<b>Chapter 30. The ZONERENAME Command</b>	<b>433</b>
Zones for SET BOUNDARY	433
Syntax	434
Operands	434
Data Sets Used	436
Usage Notes	436
Output	436
Examples	437
Example 1: Renaming an Existing Zone	437
Example 2: Creating a Target Zone from a Distribution Zone	437
Example 3: Renaming a Zone in a Copy of a CSI Data Set	438
Processing	439
Zone and Data Set Sharing Considerations	440
<b>Chapter 31. SMP/E Reports</b>	<b>441</b>
CALLLIBS Summary Report	442
Format and Explanation of Data	442
Example: CALLLIBS Summary Report for a Target Zone	443
Causer SYSMOD Summary Report	444
Format and Explanation of Data	444
Example: Causer SYSMOD Summary Report for APPLY Processing	445
CLEANUP Summary Report	446
Format and Explanation of Data	446
Example: CLEANUP Summary Report	446
Cross-Zone Requisite SYSMOD Report	447
Format and Explanation of Data	447
Example: Cross-Zone Requisite SYSMOD Report	448
Cross-Zone Summary Report	449
Format and Explanation of Data	449
Example: Cross-Zone Summary Report for APPLY Processing	452
Deleted SYSMOD Report	453
Format and Explanation of Data	453
Example: Deleted SYSMOD Report for APPLY	454
Element Summary Report	455

Format and Explanation of Data	455
Example: APPLY CHECK Element Summary Report	459
Exception SYSMOD Report	460
Format and Explanation of Data	460
Example: Exception SYSMOD Report	462
File Allocation Report	463
Format and Explanation of Data	463
Example: File Allocation Report for APPLY	466
GENERATE Summary Report	467
Format and Explanation of Data	467
Examples	468
JCLIN Cross-Reference Report	471
Format and Explanation of Data	471
Example: JCLIN Cross-Reference Report	472
JCLIN Summary Report	473
Format and Explanation of Data	473
Example: JCLIN Summary Report	476
LIST Summary Report	477
Format and Explanation of Data	477
Example: LIST Summary Report	478
MOVE/RENAME/DELETE Report	479
Format and Explanation of Data	479
Example: Report for APPLY Processing	484
RECEIVE Exception SYSMOD Data Report	485
Format and Explanation of Data	485
Examples	486
RECEIVE Summary Report	488
Format and Explanation of Data	488
Examples	490
REJECT Summary Report	492
Format and Explanation of Data	492
Examples	494
SOURCEID Report	498
Format and Explanation of Data	498
Examples	499
SYSMOD Comparison Report	500
Format and Explanation of Data	500
Example: SYSMOD Comparison Report	502
SYSMOD Regression Report	503
Format and Explanation of Data	503
Example: APPLY SYSMOD Regression Report	504
SYSMOD Status Report	505
Format and Explanation of Data	505
Example: APPLY SYSMOD Status Report	507
UNLOAD Summary Report	508
Format and Explanation of Data	508
Example: UNLOAD Summary Report	509
ZONEEDIT Summary Report	510
Format and Explanation of Data	510
Examples	511
ZONEMERGE Report	512
Format and Explanation of Data	512
Examples	513

<b>Chapter 32. SMP/E Modification Control Statements</b> . . . . .	517
++APAR MCS . . . . .	519
++ASSIGN MCS . . . . .	521
Data Element MCS . . . . .	523
Supporting Several Languages . . . . .	524
++DELETE MCS . . . . .	529
++FUNCTION MCS . . . . .	533
++HFS MCS . . . . .	535
++HOLD MCS . . . . .	540
++IF MCS . . . . .	546
++JCLIN MCS . . . . .	548
++MAC MCS . . . . .	553
++MACUPD MCS . . . . .	559
++MOD MCS . . . . .	563
++MOVE MCS . . . . .	570
++NULL MCS . . . . .	573
++PTF MCS . . . . .	574
++RELEASE MCS . . . . .	576
++RENAME MCS . . . . .	580
++SRC MCS . . . . .	582
++SRCUPD MCS . . . . .	586
++USERMOD MCS . . . . .	589
++VER MCS . . . . .	591
++ZAP MCS . . . . .	598
<b>Chapter 33. SMP/E PARMLIB Member Control Statements</b> . . . . .	603
Syntax . . . . .	604
Operands . . . . .	604
Usage Notes . . . . .	604
Examples . . . . .	605
Example 1: Defining a New OP CODE for Special Assemblers . . . . .	605
Example 2: Overriding an SMP/E-Defined OP CODE . . . . .	605
<b>Chapter 34. SMP/E Data Sets</b> . . . . .	607
Distribution Library (DLIB) . . . . .	607
INFILE Data Set . . . . .	607
Link Library (LKLIB) . . . . .	608
OUTFILE Data Set . . . . .	608
PARMLIB Data Set . . . . .	608
SMP_CNTL Data Set . . . . .	609
SMP_CSI Data Set . . . . .	609
SMP_DEBUG Data Set . . . . .	610
SMP_HOLD Data Set . . . . .	610
SMP_JCLIN Data Set . . . . .	610
SMP_LIST Data Set . . . . .	611
SMP_LOG Data Set . . . . .	611
SMP_LOGA Data Set . . . . .	612
SMP_LTS Data Set . . . . .	612
SMP_MTS Data Set . . . . .	613
SMP_OBJ Data Set . . . . .	614
SMP_OUT Data Set . . . . .	614
SMP_PTFIN Data Set . . . . .	614
SMP_PTS Data Set . . . . .	615

SMPPUNCH Data Set	615
SMPRPT Data Set	616
SMPSCDS Data Set	616
SMPSNAP Data Set	617
SMPSTS Data Set	617
SMPTLIB Data Set	618
SMPWRK1 Data Set	619
SMPWRK2 Data Set	620
SMPWRK3 Data Set	620
SMPWRK4 Data Set	621
SMPWRK6 Data Set	621
SMPnnnnn Data Set	621
SYSLIB Data Set	622
SYSPRINT Data Set	622
SYSPUNCH Data Set	623
SYSUT1, SYSUT2, and SYSUT3 Data Sets	623
SYSUT4 Data Set	624
Target Library	624
Text Library (TXLIB)	624
Zone Statement	624
<b>Chapter 35. SMP/E Data Set Entries</b>	627
How the Data Sets Are Organized	627
How Data Set Entries Are Organized	629
Entries that Control Processing	631
Entries that Define Status and Structure	633
ASSEM Entry (Distribution and Target Zone)	640
BACKUP Entries (SMPSCDS)	645
Data Element Entry (Distribution and Target Zone)	648
DDDEF Entry (Distribution, Target, and Global Zone)	654
DLIB Entry (Distribution and Target Zone)	670
DLIBZONE Entry (Distribution Zone)	675
FMIDSET Entry (Global Zone)	679
GLOBALZONE Entry (Global Zone)	682
HFS Entry (Distribution and Target Zone)	687
HOLDDATA Entry (Global Zone)	695
LMOD Entry (Distribution and Target Zone)	698
MAC Entry (Distribution and Target Zone)	715
MCS Entry (SMPPTS)	721
MOD Entry (Distribution and Target Zone)	723
MTSMAC Entry (SMPMTS)	739
OPTIONS Entry (Global Zone)	741
SRC Entry (Distribution and Target Zone)	750
STSSRC Entry (SMPSTS)	757
SYSMOD Entry (Distribution and Target Zone)	759
SYSMOD Entry (Global Zone)	774
TARGETZONE Entry (Target Zone)	782
UTILITY Entry (Global Zone)	787
ZONESET Entry (Global Zone)	794
<b>Chapter 36. SMP/E Installation-Wide Exit Routines</b>	797
Driver for Installation-Wide Exit Routines	798
RECEIVE Exit Routine	799

Retry Exit Routine . . . . .	801
<b>Chapter 37. GIMDTS: Data Transformation Service Routine . . . . .</b>	<b>803</b>
Calling GIMDTS . . . . .	804
Processing . . . . .	804
Return Codes . . . . .	804
<b>Appendix A. SMP/E Naming Conventions . . . . .</b>	<b>805</b>
Naming Conventions for HOLD Reason IDs and HOLD Classes . . . . .	807
Error Reason IDs . . . . .	807
System Reason IDs . . . . .	807
User Reason IDs . . . . .	808
Class Values . . . . .	808
Naming Conventions for Source IDs . . . . .	808
Naming Conventions for SYSMODs . . . . .	809
Function SYSMOD IDs . . . . .	810
PTF, APAR, and USERMOD SYSMOD IDs . . . . .	810
<b>Appendix B. Sample SMP/E Cataloged Procedure . . . . .</b>	<b>813</b>
Required JCL Statements . . . . .	813
JOB Statement . . . . .	813
EXEC Statement . . . . .	813
DD Statements . . . . .	815
Sample Cataloged Procedure for SMP/E . . . . .	815
How to Determine the Appropriate SYSLIB Concatenation . . . . .	818
SMP/E Storage Requirements . . . . .	821
<b>Appendix C. Dynamic Allocation . . . . .</b>	<b>823</b>
Sources of Information for Dynamic Allocation . . . . .	823
DDDEF Entries . . . . .	823
Module GIMMPDFT . . . . .	824
How Dynamic Allocation Works . . . . .	827
<b>Appendix D. Processing the SMP/E RC Operand . . . . .</b>	<b>829</b>
<b>Appendix E. Sharing SMP/E Data Sets . . . . .</b>	<b>831</b>
Types of Access . . . . .	831
Command Processing Phases . . . . .	832
Using the System Enqueue Facility . . . . .	833
Sharing the Global Zone and SMPPTS Data Set . . . . .	834
<b>Appendix F. Restricting Which Utilities Are Called . . . . .</b>	<b>835</b>
Summary of GIMUTTBL Processing . . . . .	835
Using GIMUTTBL . . . . .	835
<b>Appendix G. Summary of All SMP/E Releases . . . . .</b>	<b>839</b>
ADDIN Support Deleted . . . . .	841
Additional Element Types . . . . .	841
APAR Regression Reporting . . . . .	842
Automatic Rereceive of Reworked SYSMODs . . . . .	842
BLOCKSIZE=0 Support (CBIPO Dialogs) . . . . .	842
BLOCKSIZE=8800 for SMP/E Data Sets . . . . .	842
CBIPO Dialogs . . . . .	842

Cleanup of Target Zone Data Sets . . . . .	843
Comparing the SYSMODs Installed in Two Zones . . . . .	843
Concatenated DDDEF Entries . . . . .	843
Creating and Updating DDDEF Entries through the SMP/E Dialogs . . . . .	844
Cross-Product Load Modules for Products in Different Zones: LINK Command . . . . .	844
Cross-Product Load Modules for Products in the Same Zone: Improved Processing . . . . .	844
Cross-Zone Requisite Reporting . . . . .	845
CSECT Delete Support . . . . .	845
Data Element Changes . . . . .	845
Data Facility Product (DFP) Support . . . . .	845
Data Set Sharing . . . . .	846
Default Space Allocation for JCL Data Set: Improved Method . . . . .	846
Deleting and Superseding the Same Function . . . . .	846
Dialogs . . . . .	846
Dumps for VSAM Errors . . . . .	847
Dynamic Data Set Allocation . . . . .	847
Dynamic Dump Allocation (CBIPO Dialogs) . . . . .	848
Exception SYSMOD Control . . . . .	848
Exception SYSMOD Reporting . . . . .	849
Expanded Service Option (ESO) Support . . . . .	849
Format Change for CSI and SCDS Data Sets . . . . .	849
Format of MLPA Member (CBIPO Dialogs) . . . . .	850
Format of Tables in the CIDTABL Data Set (CBIPO Dialogs) . . . . .	850
FunctionPac Support (CBIPO Dialogs) . . . . .	850
GENERATE Command . . . . .	850
GROUP Processing Improvements . . . . .	850
Hierarchical File Systems . . . . .	851
High-Level Qualifier for RELFILE Data Sets . . . . .	851
ISPF Version 2 Release 3 and ISPF/PDF Version 2 Release 3 Required . . . . .	851
Japanese Messages and Dialogs . . . . .	851
JCLIN Reports . . . . .	852
JCLIN Saved at ACCEPT Time . . . . .	852
Job Management Function (CBIPO Dialogs) . . . . .	852
Link-Edit Automatic Library Call Support . . . . .	852
Messages Rewritten . . . . .	853
MOVE/RENAME/DELETE Processing . . . . .	854
MVSCP Elimination (CBIPO Dialogs) . . . . .	854
MVS/370 Not Supported After SMP/E Release 8.1 . . . . .	854
New Packaging . . . . .	854
Obsolete MCS Types and Operands Deleted . . . . .	854
Panel Identifier for Dialogs . . . . .	854
Performance Improvement: Automatically Using LSR . . . . .	855
Performance Improvements . . . . .	855
RACF Protection of Dynamically Allocated Data Sets . . . . .	855
RACF to Permit Read-Only Access to CSI Data Sets . . . . .	855
Regressed Subentries Deleted . . . . .	856
Receiving RELFILE Data Sets from DASD . . . . .	856
Recovery for Data Sets That Run Out of Space . . . . .	856
REJECT Improvements . . . . .	856
REPORT CALLLIBS Command . . . . .	857
Root Cause Analysis . . . . .	857
SMPLOG Improvements . . . . .	857



SMPTLIB Improvements	858
SMPWRK5 Deleted	858
SMPWRK6 Data Set Required	858
Source ID Changes	858
Source ID Exclusion	858
Source ID Report	859
Suppression of Member Names for Copy Processing	859
Symbolic Substitution in Data Set Names (CBIPO Dialogs)	859
SYSGEN No Longer Supported for SMP/E	859
SYSMOD Selection	859
SYSMOD Status Report Accuracy	860
UPGRADE Operand Deleted	861
Utility Error Isolation	861
Virtual Storage Constraint Relief	861
VSAM Data Sets	861
VS1 No Longer Supported	862
Zone Descriptions	862
Zone Editing	862
Zone Utility Commands	862
ZONESETs with Mixed Zones	863
4-Digit Device Addresses (CBIPO Dialogs)	863
<b>Appendix H. Compatibility with Previous SMP/E Releases</b>	<b>865</b>
<b>Appendix I. Performance Improvements through Local Shared Resources (LSR)</b>	<b>867</b>
SMP/E's Automatic Use of Local Shared Resources (LSR)	867
Improving APPLY and ACCEPT Processing for a Large Number of Changes	867
<b>Appendix J. GIMGNIAP: HFS Copy Utility Invocation Program</b>	<b>869</b>
Control Statements Used to Invoke GIMGNIAP	869
The INVOKE Control Statement	869
COPY Control Statement	870
SELECT Control Statement	871
Rules for GIMGNIAP Control Statements	872
Return Codes	874
JCL Statements Used in the HFSINST Job	875
<b>Appendix K. List of Macros Intended for Customer Use</b>	<b>879</b>
<b>Glossary</b>	<b>881</b>
<b>Bibliography</b>	<b>895</b>
The SMP/E Library	895
Classes and Self-Study Courses for SMP/E	897
Related Publications	898
General Interest	898
ISPF Version 4	898
ISPF and ISPF/PDF Version 3	898
ISPF and ISPF/PDF Version 2 Release 3	898
OpenEdition MVS	898
MVS/ESA SP Version 5	899
MVS/ESA SP Version 4	899

---

MVS/ESA (MVS/SP Version 3) . . . . .	899
MVS/XA (MVS/SP Version 2) . . . . .	900
MVS/370 (MVS/SP Version 1) . . . . .	900
<b>Index</b> . . . . .	<b>903</b>



---

## 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 IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the user's responsibility.

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, 208 Harbor Drive, Stamford, CT 06904-2501.

---

## Programming Interface Information

This publication is intended to help you run SMP/E commands and understand the output from SMP/E processing.

This publication also documents Product-sensitive Programming Interface and Associated Guidance Information and Diagnosis, Modification or Tuning Information provided by SMP/E.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs by an introductory statement to a chapter or section.

Diagnosis, Modification or Tuning Information is provided to help you modify SMP/E.

**Warning:** Do not use this Diagnosis, Modification or Tuning Information as a programming interface.

Diagnosis, Modification or Tuning Information is identified where it occurs by an introductory statement to a chapter or section.

---

## Trademarks

The following terms, denoted by an asterisk (\*) in this publication, are trademarks of the IBM Corporation in the United States or other countries:

ACF/VTAM  
BookManager  
CBIPO  
CBPDO  
CICS  
DB2  
DFSMS  
DFSMS/MVS  
FunctionPac

GDDM  
IBM  
MVS/DFP  
MVS/ESA  
MVS/SP  
MVS/XA  
OpenEdition  
RACF  
System/370

---

## About This Book

Use this publication when you need to:

- Code SMP/E commands and modification control statements
- Read SMP/E reports
- Call SMP/E installation-wide exit routines or service routines
- Allocate SMPCSI data sets
- Set up an SMP/E cataloged procedure
- Define which utility programs SMP/E can call

---

## What This Publication Contains

The chart below shows where to find various types of information contained in this publication.

Chapter	Subject	Description
Chapter 1	Syntax Notation and Rules	Describes the syntax notations and the general syntax rules for SMP/E commands and input.
Chapter 2 through Chapter 30	Commands	Contain detailed information about the SMP/E commands, including syntax, operands, required data sets, usage notes, examples, and processing.
Chapter 31	Reports	Contains descriptions of all the reports produced by SMP/E.
Chapter 32 and Chapter 33	Control Statements	Describes modification control statements (Chapter 32). This can help you build user modifications for IBM code or package your own programs. It can also help you debug a problem caused by an error in the construction of the SYSMOD.  Describes how to code members in SYS1.PARMLIB that define macros and assembler operation codes (Chapter 33).
Chapter 34 and Chapter 35	Data Sets	Describes data sets used during SMP/E processing (Chapter 34).  Describes the entries in SMP/E data sets (Chapter 35).
Chapter 36	Installation-Wide Exit Routines	Describes the available SMP/E installation-wide exit routines and contains instructions for writing installation-wide exit routines.
Chapter 37	GIMDTS Service Routine	Describes how to use GIMDTS, the SMP/E service routine that transforms data elements so that they can be packaged inline.

This publication also includes these appendixes:

- Appendix A describes the naming conventions for the information you provide to SMP/E.
- Appendix B describes how to set up a standard SMP/E cataloged procedure.
- Appendix C describes SMP/E dynamic allocation and how to use it.
- Appendix D explains the RC operand.
- Appendix E describes how to control access to SMP/E zones.
- Appendix F explains how to define which utility programs SMP/E may call.
- Appendix G summarizes the changes added in each release of SMP/E.
- Appendix H summarizes compatibility between releases of SMP/E.
- Appendix I describes how SMP/E performance is improved through the use of local shared resources (LSR).
- Appendix J describes GIMGNIAP, which is the program that invokes the HFS copy utility.
- Appendix K lists the SMP/E macros intended for customer use.



---

## Summary of Changes

This section describes changes to SMP/E, as well as changes to the contents and organization of the SMP/E documentation.

---

### SMP/E Release 8.1 (September 1994)

This section summarizes the changes for SMP/E Release 8.1.

- **Support for MVS/ESA SP Version 5.** The CBIPO\* dialogs in SMP/E Release 8.1 have been updated to support MVS/ESA SP Version 5.
  - **MVSCP Elimination.** For CBIPO orders containing MVS/ESA SP Version 5 and later, the MVSCP options of the CBIPO dialogs are not allowed; only the IODF path is supported.

If you have an MVSCP deck (or a combined MVSCP/IOCP deck) and are installing an order at CBIPO spin level 94B or later that uses IODF data sets for its I/O definitions, the CBIPO dialogs prompt you for information about the existing I/O definitions, so a batch job can be created to migrate the I/O configuration information to an IODF data set.
  - **4-Digit Device Addresses.** The Build option of the CBIPO dialogs allows 4-digit device addresses to be entered wherever device addresses can be specified. If the address has not been genned, it is rejected.

If the order being processed contains MVS/ESA SP Version 5, the Tailor option of the CBIPO dialogs allows 4-digit device addresses to be entered on the stand-alone dump panels. If the order contains a lower level of MVS, 4-digit device addresses are not allowed.
  - **Symbolic Substitution in Data Set Names.** The Tailor and Build options of the CBIPO dialogs now support the specification and resolution of data set names that use the &SYSNAME symbol. PARMLIB support has been changed to allow the &SYSNAME symbol in data set names for LOGREC, SMF data sets, STGINDX, and PAGE data sets. This support for the &SYSNAME symbol in data set names facilitates the sharing of a master catalog in a parallel sysplex environment, thus easing the management of multiple images in that environment.

**Migration Note:** An MVS/ESA SP Version 5 redistribution order that has been tailored using symbols cannot be installed by the CBIPO dialogs in SMP/E Release 8 or earlier. This redistribution order is installable only on a system running SMP/E Release 8.1.
  - **Dynamic Dump Allocation.** The PARMLIB/PROCLIB portion of the Tailor options of the CBIPO dialogs facilitates the exploitation of the dynamic dump allocation support provided in MVS/ESA SP Version 5. If the dialogs are processing an order containing MVS/ESA SP Version 5, the user can specify where the dump information is to be written (via SMS classes or a group of DASD volumes), how the dump data sets are to be named, and whether automatic allocation of dump data sets is to be enabled. The dialogs use this information to create the necessary DUMPDS commands in the user-specified COMMNDxx PARMLIB member.

---

**Note:** The dialogs continue to support preallocated dump data sets. In addition, dynamically allocated dump data sets can coexist with pre-allocated dump data sets.

Configurations that share resources, such as a master catalog, should make use of this support instead of preallocating dump data sets. Dynamic dump allocation can help ensure that the names of dump data sets sharing a master catalog are unique within a parallel sysplex.

- **Support for OpenEdition MVS.** In support of OpenEdition\* MVS, the PARM operand has been added to the ++HFS statement so that the hierarchical file system file access bits (permission bits) can be set at installation instead of as a postinstallation task. To take advantage of this support, the SMP/E user installing a SYSMOD containing this change must have the proper authority to set access bits through the mechanism employed by BPXCOPY.
- **BLKSIZE=8800 Now Used for SMP/E Data Sets.** For the RELFILE data sets, target libraries, and distribution libraries containing SMP/E Release 8.1, data sets that are allocated with RECFM=FB and LRECL=80 are now allocated with BLKSIZE=8800.
- **New Format for CBIPO Dialog Tables.** The CBIPO dialogs now require the tables in the CIDTABL data set to be in a new format. The tables for CBIPOs at spin level 94A or later are in the required format, but tables for pre-94A CBIPOs need to be migrated to the new format. To help users with existing CIDTABL data sets that contain tables for pre-94A CBIPOs, the CIDCGCVT CLIST is provided in the SMP/E CLIST data set (GIM.SGIMCLS0) to migrate those tables to the new format. For details on using this CLIST to migrate the CIDTABL tables, see the *SMP/E R8.1 CBIPO Dialogs User's Guide*.
- **Changes to the SMP/E Library.** A new book has been added to the SMP/E library. *Migrating to the CBIPO Dialogs*, GC23-3810, helps CBIPO users migrate from using the batch installation method (which is no longer available) to using the CBIPO dialogs provided in SMP/E. In addition, the tutorial for the CBIPO dialogs contains a summary of the information in that book.
- **Last Release to Support MVS/370.** SMP/E Release 8.1 is the last release of SMP/E to run on MVS/370.

## Changes for SC28-1107-13

A new appendix, Appendix H, "Compatibility with Previous SMP/E Releases" on page 865, has been added to summarize restrictions on processing SMP/E entries and other data with earlier releases of SMP/E.

---

## SMP/E Release 8 (March 1994)

This section summarizes the changes for SMP/E Release 8.

- **Link-Edit Automatic Library Call Support.** SMP/E now allows load modules to contain modules from multiple products without explicitly specifying those modules on INCLUDE statements in link-edit steps. This implicit inclusion of modules, which is done through SYSLIB DD statements in JCLIN link-edit steps, can be useful for products that:
  - Are written in a high-level language and, as a result, include modules from libraries (such as compiler libraries) that are owned by a different product



- 
- Make use of a callable-services interface provided by another product
  - Need to include stub routines or interface modules from different products that may reside in other zones

To support such products, SMP/E can now use the CALL option and a specified SYSLIB allocation when invoking a link-edit utility. This allows the link-edit utility to include the necessary modules, reducing the need for postinstallation steps that must be run outside of SMP/E to accomplish the same task.

The LMOD entry can now contain a new subentry list (CALLLIBS) to record the SYSLIB allocation specified for a load module.

**Migration Note:** Target zones containing LMOD entries updated by SMP/E Release 8 cannot be used by SMP/E Release 7 or earlier, because the structure of the LMOD entries has changed to support the new CALLLIBS subentry list. LMOD entries are typically updated when JCLIN that defined the load module is processed.

When SMP/E services a load module built using the link-edit automatic library call feature, it rebuilds the load module without using the old load module as input. Instead, SMP/E uses as input a second load module that includes only the explicitly-defined modules from the original load module. This allows the automatic library call option to implicitly include modules at their latest service level.

SMP/E's link-edit support for the link-edit automatic library call feature is different from the SMP/E LINK command in two major respects:

- **Defining the modules to be included.** The automatic library call option is best used when a load module must include many modules from other products, and it is difficult and error-prone (and perhaps impossible) to define all the modules to be included.

The LINK command is more appropriate when a load module needs to include only a few specific modules from other products.

- **Servicing load modules containing modules from other products.** When a load module is built using the automatic library call option, SMP/E cannot completely service the load module because it does not know the full content of the load module. Specifically, the load module is not automatically rebuilt when an implicitly-included module is serviced. However, the new REPORT CALLLIBS command can be used to identify and relink such load modules. REPORT CALLLIBS is described in more detail below.

When a load module is updated with the LINK command, SMP/E knows the complete content of the load module and can automatically maintain the load module during subsequent APPLY and RESTORE processing.

- **New REPORT CALLLIBS Command.** The REPORT command can now be used to identify and relink load modules when implicitly-included modules in a particular library are updated. Specifically, the REPORT CALLLIBS command tells you which load modules have a CALLLIBS subentry list in their LMOD entry (and therefore use the automatic library call option to implicitly include modules from a specified library). If requested, REPORT CALLLIBS also creates jobs to relink the load modules that are reported on. A separate job is created for each affected zone.

---

This command is helpful after a particular compiler library or callable-service data set has been upgraded with service or a new release. By specifying the upgraded library on the REPORT CALLLIBS command, you can find out which load modules for other products implicitly include modules from the upgraded library. By running the jobs created by REPORT CALLLIBS, you can relink these load modules so they include the latest routines from the upgraded library.

- **Recovery for Data Sets That Run Out of Space.** Recovery processing for data sets that have run out of space has been improved in the following ways:
  - **Extended retry processing.** When several members of a given data set need to be updated, SMP/E often “batches” those updates in a single utility call. When the utility attempts to update those members, the data set may run out of space (experience an x37 abend). In this case, if the user has requested retry processing, SMP/E compresses the data set and retries the utility.

In the past, if compressing a data set that had run out of space did not reclaim enough space, SMP/E terminated the job. Now, if compression fails to remedy the problem, SMP/E “debatches” the updates for that data set and retries the utility separately for each member in the data set, compressing the data set again if necessary. If this final attempt fails, the resulting x37 abend is treated as an unacceptable utility return code, and processing continues for other eligible updates.

- **Excluding specific data sets from retry processing.** It is now simpler for you to request retry processing when you want all but a few specific data sets to be included. In the past, the only way to do this was to specify each desired data set as a RETRYDDN value in the OPTIONS entry. Now, you can achieve the same result by using the new EXRTYDD subentry for the OPTIONS entry along with RETRYDDN(ALL). Follow these steps:
  1. Specify RETRYDDN(ALL) to include all your data sets in retry processing.
  2. Specify EXRTYDD and the ddnames of the specific data sets to be excluded from retry processing.

**Migration Note:** OPTIONS entries containing the new EXRTYDD subentry cannot be used by SMP/E Release 7 or earlier.

- **Support for Relative File Data Sets.** Support for data sets in relative files (RELFILEs) has been expanded.
  - **High-level qualifier for RELFILE data sets.** SMP/E now supports the use of a single, common, high-level qualifier for SYSMODs packaged in RELFILE format, instead of using the ID of the SYSMOD as the high-level qualifier. In the past, names of RELFILE data sets were in the format *sysmodid.Fn*. The names of RELFILE data sets using this new high-level qualifier will be in the format *hlq.sysmodid.Fn*.

This support reduces the number of high-level qualifiers that must be defined to RACF\* (or another security-management program), as well as the need to extend security management definitions each time new products are to be installed.

---

**Note:** The naming convention used for SMPTLIB data sets allocated by SMP/E remains unchanged—those data set names do **not** include the new high-level qualifier used for RELFILE data sets.

- **Receiving RELFILE data sets from DASD.** In the past, the SMP/E RECEIVE command could process RELFILE data sets only if they were on tape. Now, SMP/E can process RELFILE data sets located on DASD as well.

- **Support for Hierarchical File Systems.** SMP/E can now install and maintain elements and load modules residing in a *hierarchical file system (HFS)*. Here are some highlights of this support:

- DDDEF entries can be used to allocate paths within a hierarchical file system.
- A new type of modification control statement, ++HFS, is provided for packaging elements residing in a hierarchical file system.
- Processing of SMP/E commands has been changed to provide additional support. This includes changes to JCLIN processing of link-edit steps for load modules residing in a hierarchical file system.
- A new type of utility, the HFS copy utility, is recognized by SMP/E. The associated UTILITY entry is pointed to by the HFSCOPY subentry in the OPTIONS entry.

**Migration Note:** OPTIONS entries containing the new HFSCOPY subentry cannot be used by SMP/E Release 7 or earlier.

- **Support for Expanded Service Option (ESO).** Where available, the Expanded Service Option (ESO) has replaced the program update tape (PUT) process as the vehicle for delivering preventive service. SMP/E has replaced its support for PUTs (including the format of PUT tapes) with support for ESO tapes.

- **Suppression of Member Names for Copy Processing.** Through the new LIST subentry in the UTILITY entry, SMP/E allows you to suppress the listing of member names during all SMP/E processing that invokes the copy or compress utility. With this new subentry, you can control the generation of the LIST=NO operand specification on the COPY control statement and eliminate unwanted output from invocations of the copy or compress utility.

**Migration Note:** UTILITY entries containing the new LIST subentry cannot be used by SMP/E Release 7 or earlier.

- **Changes to the SMP/E Library.** Two new books have been added to the SMP/E library. They will be available at general availability of SMP/E Release 8.

- The *SMP/E R8 Primer*, GC23-3771, introduces the basic principles people need to understand in order to use SMP/E to install and maintain software, without the SMP/E “jargon” and expert-level details found in other SMP/E publications.
- The *SMP/E R8 Master Index*, GC23-3812, helps users quickly determine which book in the SMP/E library contains the information they are looking for.

---

In addition, information from two separate packaging books has been combined in a single publication. Specifically, the *SMP/E Program Packaging Guide*, SC23-0221, has been merged into the *Standard Packaging Rules for MVS-Based Products* manual, SC23-3695. The resulting book can be used by any non-IBM developer to package products and service, regardless of whether the software is to be distributed through IBM.



---

## Chapter 1. Syntax Notation and Rules

This chapter explains the syntax notation and rules for SMP/E commands, SMP/E modification control statements (MCSs), and control statements in members of the PARMLIB data set used by SMP/E. It describes:

- How to read the notation used to show how commands and control statements should be coded
- The rules to follow when coding commands and control statements

---

### How to Read the Syntax Diagrams

Throughout this publication, the structure defined below is used in describing syntax:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

The ► symbol indicates the beginning of a statement.

The → symbol indicates that the statement syntax is continued on the next line.

The ◀ symbol indicates that a statement is continued from the preceding line.

The ◀◀ symbol indicates the end of a statement.

- Required items appear on the horizontal line (main path).

►►—STATEMENT—required\_item—◀◀

- Optional items appear below the main path.

►►—STATEMENT—  
                  └ optional\_item ┘—◀◀

- If you can choose from two or more items, they appear in a vertical stack.

If you **must** choose one of the items, one item of the stack appears on the main path.

►►—STATEMENT—  
                  └ required\_choice1 ┘—◀◀  
                  └ required\_choice2 ┘

If choosing one of the items is optional, the entire stack appears below the main path.

►►—STATEMENT—  
                  └ optional\_choice1 ┘—◀◀  
                  └ optional\_choice2 ┘

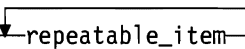
If one of the optional items is the default, it appears above the main path and the remaining choices will be shown below.

►►—STATEMENT—  
                  └ default\_choice1 ┘—◀◀  
                  └ optional\_choice2 ┘  
                  └ optional\_choice3 ┘

- Keywords appear in uppercase (for example, PARM1). **They must be spelled exactly as shown.**
- Variables appear in lowercase italics (for example, *parmx*). They represent user-supplied names or values.

▶▶—STATEMENT—*variable*—▶▶

- An arrow returning to the left above the main line indicates an item that can be repeated.

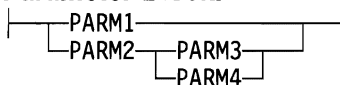
▶▶—STATEMENT——▶▶

A repeat arrow above a stack indicates that you can make more than one choice from the stacked items, or repeat a single choice.

- A repeat arrow above a stack of keywords means that you can enter one or more of the keywords. However, **each keyword can be entered only once.**
- A repeat arrow above a variable means that you can enter one or more values for the variable. However, **each value can be entered only once.**
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.
- Sometimes a single substitution represents a set of several parameters. For example, in the following diagram, the callout Parameter Block can be replaced by any of the interpretations of the subdiagram that is labeled Parameter Block:.

▶▶—STATEMENT——▶▶

### Parameter Block:



---

## Syntax Rules

Follow these rules when you code SMP/E commands, MCSs, and PARMLIB member control statements:

- Use uppercase letters to enter all SMP/E input other than the text within a comment.
- Start each statement on a new logical 80-byte record.

For SMP/E commands, enter the command name first, followed by any operands.

For MCSs, do the following:

- Code the “++” for the MCS in columns 1 and 2.
- Code the MCS name on the same line as the “++”.

For PARMLIB member control statements, do the following:

- Code the KEY=xxx operand first.

- Do **not** continue PARMLIB member control statements on a subsequent record.

**Note:** Except for these restrictions, the SMP/E commands, MCSs, and PARMLIB member control statements can begin and end anywhere up to and including column 72.

- You can code optional information in any order, except where noted in the syntax and operand descriptions.
- Include at least one blank between each operand.
- Separate operands and their values with a blank or comma.
- You can continue a command or statement on more than one line. SMP/E assumes a statement is continued if it did not find a period (.) before column 73.

**Note:** PARMLIB members are an exception—they cannot span multiple records.

- Start comments with “/\*” and end them with “\*/”. A comment can appear anywhere on a statement, but should not begin in column 1. (When “/\*” starts in column 1, it indicates the end of an input data set.) A comment after the ending period **must** start on the same line as the period. You cannot specify any additional operands or comments after that final comment. For example, you can code a comment like this:

```
SET      BDY(MVSTST1)      /* Comment after period
                           continued on subsequent
                           records is ok.          */.
```

However, you should **not** code a comment like this:

```
SET      BDY(MVSTST1)      /* Comment after period   */
                           /* ok but next comment     */
                           /* will give syntax error. */.
```

This causes a syntax error at the start of the second comment after the period.

- Comments can be in single-byte characters (such as English alphanumeric characters) or in double-byte characters (such as Kanji).
- End each statement with a period.
- SMP/E completes processing for one statement before it starts processing the next one.
- SMP/E ignores columns 73 through 80. If data, such as a period, is specified beyond column 72, SMP/E ignores it and indicates an error in the statement after the one containing that data.





---

## Chapter 2. The ACCEPT Command

The ACCEPT command is used to cause SMP/E to install the elements supplied by a SYSMOD into the distribution libraries (or DLIBs). The ACCEPT process:

- Selects SYSMODs present in the global zone that are applicable to the specified distribution libraries
- Makes sure all other required SYSMODs have been accepted or are being accepted concurrently
- Selects the elements from the accepted SYSMODs based on the functional and service level of those elements in the distribution libraries and the relationship between the SYSMODs being installed, ensuring that no current service is regressed by the installation of another SYSMOD
- Calls system utilities to install the elements into the distribution libraries
- Records the functional and service levels of the new elements in the distribution zone
- Records the installation of the SYSMOD in the distribution zone
- Deletes the global zone SYSMOD and PTS MCS entries for those SYSMODs successfully processed

The ACCEPT process is controlled by:

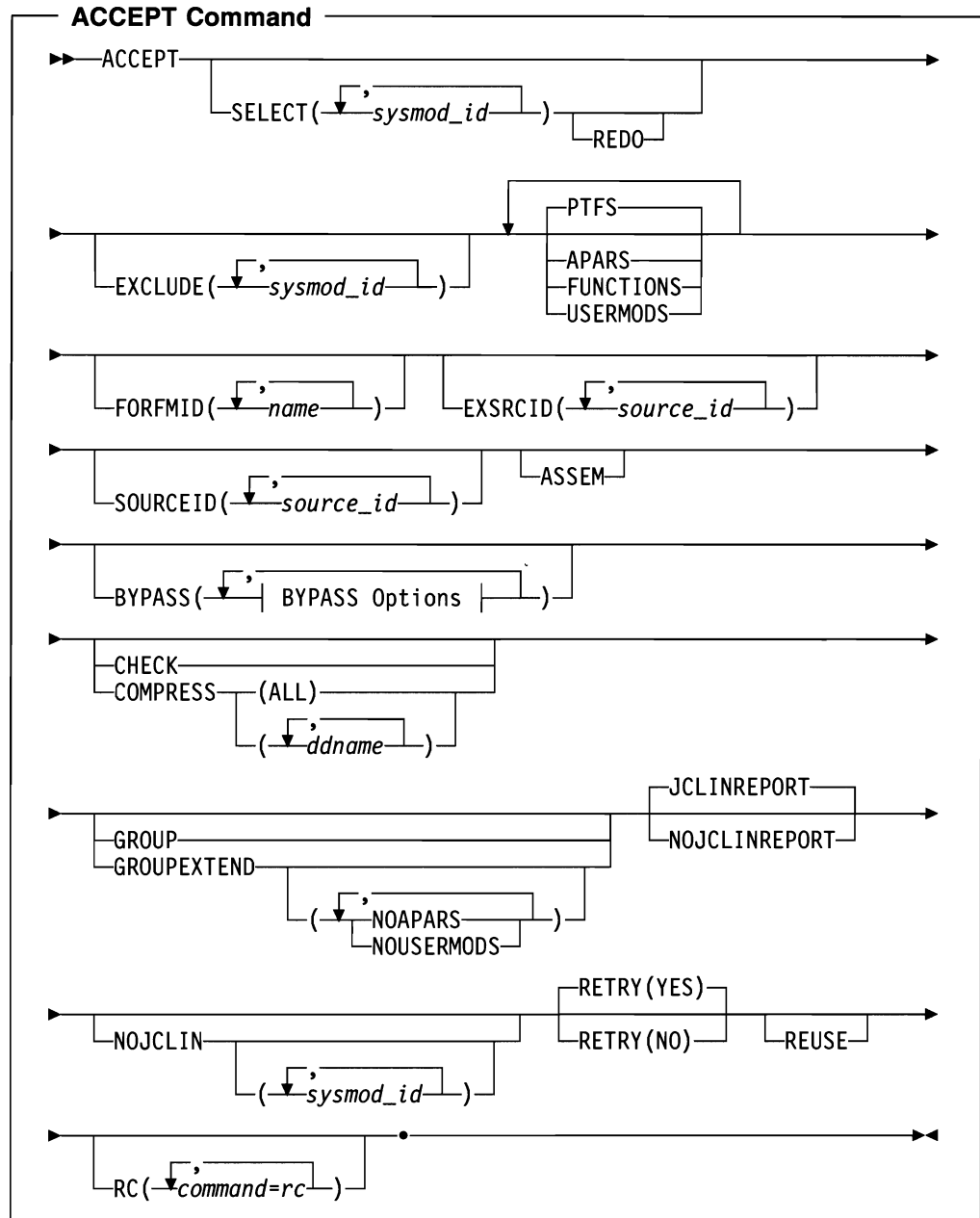
- The information in the distribution zone reflecting the status and structure of the distribution libraries
- Information on the SYSMODs indicating their applicability
- Information in the OPTIONS and UTILITY entries
- Operands on the ACCEPT command

---

### Zones for SET BOUNDARY

For the ACCEPT command, the SET BOUNDARY command must specify the distribution zone associated with the distribution libraries where the SYSMODs will be installed.

Syntax





**BYPASS**

You can specify any of these options:

APPLYCHECK  
 HOLDCLASS  
 HOLDERROR  
 HOLDSYSTEM  
 HOLDUSER  
 ID  
 IFREQ  
 PRE  
 REQ

**Note:** If you specify both **BYPASS** and **GROUPEXTEND**, SMP/E does not include superseding SYSMODs needed to take the place of requisites that have been bypassed.

During **CHECK** processing, if you want to see whether any superseding SYSMODs are available for requisites that have been bypassed, specify **GROUPEXTEND** without **BYPASS**.

**BYPASS(APPLYCHECK)**

indicates that SYSMODs should be accepted even if they have not been applied. For example, if you are preparing the distribution libraries before doing a system generation, you want to accept SYSMODs that have not been applied.

**Note:** **APPLYCHECK** can also be specified as **APPCHK**.

**BYPASS(HOLDCLASS(value,...))**

indicates that exception SYSMODs associated with the specified class names should not be held. The list of class names is required.

These are the hold classes you can specify:

**Class Explanation**

**ERREL** The SYSMOD is held for an error reason ID but should be installed anyway. IBM has determined that the problem the SYSMOD resolves is significantly more critical than the error reflected by the holding APAR.

**UCLREL** UCLIN needed for the SYSMOD has been handled by IBM and no longer requires your attention.

**BYPASS(HOLDERROR)**

indicates that exception SYSMODs associated with the specified error reason IDs should not be held. The list of reason IDs is optional. If you include one, only the reason IDs specified on it are bypassed. If you do not include a list, all error reason IDs are bypassed.

**Note:** **HOLDERROR** can also be specified as **HOLDERR**.

**BYPASS(HOLDSYSTEM)**

indicates that exception SYSMODs associated with the specified system reason IDs should not be held. The list of reason IDs is optional. Generally, you should specify **BYPASS(HOLDSYSTEM)** on all **ACCEPT CHECK** commands, and **BYPASS(HOLDSYSTEM(reason\_id,...))** on all **ACCEPT**

commands for all system reason IDs for which appropriate action has been (or will be) taken.

If you include a list of reason IDs, all the SYSMODs with the specified reason IDs are bypassed. You should, therefore, make sure the appropriate action has been taken for all these SYSMODs. If you do not include a list, all system reason IDs are bypassed.

**Note:** **HOLDSYSTEM** can also be specified as **HOLDSYS**.

These are the system reason IDs currently used by IBM:

ID	Explanation
<b>ACTION</b>	The SYSMOD needs special handling before or during APPLY processing, ACCEPT processing, or both.
<b>AO</b>	The SYSMOD may require action to change automated operations procedures and associated data sets and user exits in products or in customer applications. The PTF cover letter describes any changes (such as to operator message text, operator command syntax, or expected actions for operator messages and commands) that can affect automation routines.
<b>DELETE</b>	The SYSMOD contains a ++DELETE MCS, which deletes a load module from the system.
<b>DEP</b>	The SYSMOD has a software dependency.
<b>DOC</b>	The SYSMOD has a documentation change that should be read before the SYSMOD is installed.
<b>EC</b>	The SYSMOD needs a related engineering change.
<b>EXRF</b>	The SYSMOD must be installed in both the active and the alternative MVS/XA* Extended Recovery Facility (XRF) systems at the same time to maintain system compatibility. (If you are not running XRF, you should bypass this reason ID.)
<b>FULLGEN</b>	The SYSMOD needs a complete system or subsystem generation to take effect.
<b>IOGEN</b>	The SYSMOD needs a system or subsystem I/O generation to take effect.
<b>MSGSKEL</b>	This SYSMOD contains message changes that must be compiled for translated versions of the message changes to become operational on extended TSO consoles.  If you want to use translated versions of the messages, you must run the message compiler once for the library containing the English message outlines, and once for each additional language you want to be available on your system. For details, see the <i>MVS/ESA Planning: Operations</i> manual.  If you want to use <b>only</b> the English version of the messages, you do not need to run the message compiler. You should bypass this reason ID.
<b>MVSCP</b>	The SYSMOD requires the MVS configuration program to be run for the change to take effect.

### **BYPASS(HOLDUSER)**

indicates that exception SYSMODs associated with the specified user reason IDs should not be held. The list of reason IDs is optional. If you include one, only the reason IDs specified on it are bypassed. If you do not include a list, all user reason IDs are bypassed.

### **BYPASS(ID)**

indicates that SMP/E should ignore any errors it detects when checking the SYSMOD's RMID and UMIDs. Generally, you should specify **BYPASS (ID)** on all ACCEPT CHECK commands to check for possible regressions.

### **BYPASS(IFREQ)**

indicates that SMP/E should ignore any conditional requisites that are missing. Generally, you should specify **BYPASS (IFREQ)** on all ACCEPT CHECK commands to check for possible regressions.

### **BYPASS(PRE)**

indicates that SMP/E should ignore any prerequisites that are missing. Generally, you should specify **BYPASS (PRE)** on all ACCEPT CHECK commands to check for possible regressions.

### **BYPASS(REQ)**

indicates that SMP/E should ignore any requisites that are missing. Generally, you should specify **BYPASS (REQ)** on all ACCEPT CHECK commands to check for possible regressions.

### **CHECK**

indicates that SMP/E should not actually update any libraries. Rather, it should just do the following:

- Test for errors other than those that could occur when the libraries are actually updated.
- Report on which libraries are affected.
- Report on any SYSMOD that would be regressed.

### **COMPRESS**

indicates which distribution libraries should be compressed.

- If you specify **ALL**, any libraries in which elements will be installed by this ACCEPT command are compressed.
- If you specify particular ddnames, those libraries are compressed regardless of whether they will be updated.

#### **Notes:**

1. **COMPRESS** can also be specified as **C**.
2. If you specify both **COMPRESS** and **CHECK**, **COMPRESS** is ignored. This is because SMP/E does not update any data sets for **CHECK**.

**EXCLUDE**

specifies one or more SYSMODs that should not be accepted.

**Notes:**

1. **EXCLUDE** can also be specified as **E**.
2. If a SYSMOD is specified on the **EXCLUDE** operand, SMP/E does **not** include this SYSMOD, even though it might be specified on the **GROUP** or **GROUPEXTEND** operand.

**EXSRCID**

indicates that SYSMODs associated with the specified source IDs should **not** be accepted.

**Notes:**

1. There are two ways to specify source IDs:
  - Explicitly, by specifying a particular source ID (for example, **PUT9303**). In this case, only that particular source ID is used.
  - Implicitly, by specifying either **\*** or **c\*** (for example, **PUT\***), where *c* is a 1- to 7-character string. In the second case, all source IDs beginning with the specified character string are used.
2. A given source ID can be explicitly specified **only once** on the **EXSRCID** operand.
3. The same source ID **cannot** be explicitly specified on both the **EXSRCID** and **SOURCEID** operands.
4. If a source ID is specified implicitly or explicitly on the **EXSRCID** operand and is also specified either implicitly or explicitly on the **SOURCEID** operand, all SYSMODs with that source ID are excluded from processing.
5. If a given SYSMOD has multiple source IDs and at least one of those source IDs is specified either implicitly or explicitly on the **SOURCEID** operand, the SYSMOD is excluded from processing if another one of its source IDs is specified either explicitly or implicitly on the **EXSRCID** operand.  
  
For example, assume PTF UZ12345 has been assigned source IDs **SMCREC** and **PUT9303**. If you specify **SOURCEID(SMC\*)** and **EXSRCID(PUT9303)**, the SYSMOD is excluded from processing.
6. If a SYSMOD that would have been included by the **GROUP** or **GROUPEXTEND** operand is excluded by the **EXSRCID** operand, SMP/E does not include it.
7. If no SYSMOD types are specified, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.

**FORFMID**

indicates that only SYSMODs for the specified FMIDs or FMIDSETs should be accepted.

**Notes:**

1. Functions containing a ++VER DELETE statement are not automatically included by the FORFMID operand. You must specify them on the SELECT operand.
2. If no SYSMOD types are specified, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.

**FUNCTIONS**

indicates that all eligible functions should be accepted.

**Notes:**

1. **FUNCTIONS** can also be specified as **FUNCTION**.
2. If **FUNCTIONS** is specified along with **SELECT**, all eligible functions are included in addition to the SYSMODs specified on **SELECT**.
3. If **FUNCTIONS** is specified along with **SOURCEID**, all functions associated with the specified source IDs are included.
4. Functions that contain a ++VER DELETE statement are not automatically included by the **FUNCTIONS** operand. You must specify them on the **SELECT** operand.

**GROUP**

indicates that if any SYSMODs specifically defined as requisites for eligible SYSMODs have not yet been accepted, SMP/E should automatically include them.

**Notes:**

1. **GROUP** can also be specified as **G**.
2. **GROUP** is mutually exclusive with **GROUPEXTEND**.
3. **GROUP** might include SYSMODs at a higher service level than the level specified by the **SOURCEID** operand.
4. If you specify **GROUP** without any other SYSMOD selection operands (such as a SYSMOD type, **SOURCEID**, **FORFMID**, or **SELECT**), **GROUP** is ignored.
5. Processing done for SYSMODs specified on the **SELECT** operand is not necessarily done for SYSMODs included by the **GROUP** operand. For example, if **REDO** is specified, only SYSMODs specified on the **SELECT** operand can be reaccepted; SYSMODs included by the **GROUP** operand are not.
6. Functions containing a ++VER DELETE statement are not automatically included by the **GROUP** operand. You must specify them on the **SELECT** operand.
7. If a SYSMOD that would have been included by the **GROUP** operand is excluded by the **EXCLUDE** or **EXSRCID** operand, SMP/E does not include it.

**GROUPEXTEND**

indicates that if a SYSMOD specifically defined as a requisite for an eligible SYSMOD has not been accepted and cannot be processed for one of the reasons shown below, SMP/E should automatically include a superseding



SYSMOD. Table 1 on page 13 shows what GROUPEXTEND includes, depending on why the requisite cannot be processed.

For a Requisite That Is:	GROUPEXTEND Includes:
<ul style="list-style-type: none"> <li>Held for an error reason ID</li> </ul>	<ul style="list-style-type: none"> <li>A SYSMOD that supersedes the requisite or</li> <li>A SYSMOD that matches or supersedes the error reason ID</li> </ul>
<p>One of the following:</p> <ul style="list-style-type: none"> <li>Held for a system reason ID</li> <li>Held for a user reason ID</li> <li>Accepted in error</li> <li>Not available</li> </ul>	<ul style="list-style-type: none"> <li>A SYSMOD that supersedes the requisite</li> </ul>

You can specify **NOAPARS** or **NOUSERMODS** (or both) to limit the types of SYSMODs that are included by GROUPEXTEND to resolve error reason IDs. The default is to include all eligible SYSMODs, regardless of SYSMOD type.

#### **NOAPARS**

indicates that SMP/E should exclude APARs that resolve error reason IDs.

#### **NOUSERMODS**

indicates that SMP/E should exclude USERMODs that resolve error reason IDs.

#### **Notes:**

- GROUPEXTEND** can also be specified as **GEXT**.
- GROUPEXTEND** is mutually exclusive with **GROUP**.
- If you specify both **BYPASS** and **GROUPEXTEND**, SMP/E does not include any superseding SYSMODs needed to take the place of requisites that have been bypassed.  
  
During CHECK processing, if you want to see whether any superseding SYSMODs are available for requisites that have been bypassed, specify **GROUPEXTEND** without **BYPASS**.
- GROUPEXTEND** might include SYSMODs at a service level higher than that specified by the **SELECT** or **SOURCEID** operand.
- Functions and excluded SYSMODs are not automatically included by **GROUPEXTEND**.
- Processing done for SYSMODs specified on the **SELECT** operand is not necessarily done for SYSMODs specified by the **GROUPEXTEND** operand. For example, if **REDO** is specified, only SYSMODs specified on the **SELECT** operand are reaccepted; SYSMODs included by the **GROUPEXTEND** operand are not.

7. If a SYSMOD that would have been included by the GROUPEXTEND operand is excluded by the EXCLUDE or EXSRCID operand, SMP/E does not include it.
8. When **GROUPEXTEND** is specified, SMP/E examines more SYSMODs than it does if **GROUP** were specified. Because of this additional processing, the ACCEPT command runs longer than if **GROUP** was specified, and a larger region size may be needed. On the other hand, GROUPEXTEND reduces the amount of time you would otherwise spend searching for missing requisites.

### JCLINREPORT

indicates that SMP/E is to write the JCLIN reports after processing inline JCLIN. This is the default when inline JCLIN is processed at ACCEPT time (ACCJCLIN is set in the DLIBZONE entry).

**Note:** JCLINREPORT can also be specified as JCLR.

### NOJCLIN

indicates that SMP/E should not process inline JCLIN for the specified SYSMODs. For example, if you are reaccepting SYSMODs, you may not want to process inline JCLIN that would change distribution zone entries that should not be changed.

If you include a list of SYSMOD IDs, SMP/E skips JCLIN processing only for the specified SYSMODs. If you do not include a list of SYSMOD IDs, SMP/E skips JCLIN processing for all SYSMODs.

**Note:** If inline JCLIN is not being processed at ACCEPT time (ACCJCLIN is not set in the DLIBZONE entry), you do not need to specify **NOJCLIN**.

### NOJCLINREPORT

indicates that SMP/E should not write any JCLIN reports after processing inline JCLIN.

**Note:** NOJCLINREPORT can also be specified as NOJCLR.

### PTFS

indicates that all eligible PTFs should be accepted.

#### Notes:

1. PTFS can also be specified as PTF.
2. PTFS is the default SYSMOD type for mass-mode processing. If no other SYSMOD types are specified, only PTFs are processed, even if PTFS was not specified.
3. If PTFS is specified along with **SELECT**, all eligible PTFs are included in addition to the SYSMODs specified on **SELECT**.
4. If PTFS is specified along with **SOURCEID**, all PTFs associated with the specified source IDs are included.

### RC

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ACCEPT command.

Before SMP/E processes the ACCEPT command, it checks whether the return codes for the specified commands are less than or equal to the values speci-

fied on the RC operand. If so, SMP/E can process the ACCEPT command. Otherwise, the ACCEPT command fails. For more information about the RC operand, see Appendix D.

**Notes:**

1. The RC operand **must be the last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ACCEPT command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

**REDO**

indicates that if any SYSMOD specified on SELECT has already been successfully accepted, it should be reaccepted.

**Notes:**

1. If you specify **REDO**, you must also specify **SELECT**.
2. If **GROUP** or **GROUPEXTEND** is also specified, REDO does not reaccept SYSMODs included by the GROUP or GROUPEXTEND operand. It only processes SYSMODs specified on the SELECT operand.

**RETRY**

indicates whether SMP/E should try to recover from out-of-space errors for utilities it calls.

**YES**

indicates that SMP/E should try to recover and should retry the utility if a RETRYDDN list is available in the OPTIONS entry that is in effect. RETRY(YES) is the default.

If retry processing does not reclaim sufficient space and input to the utility was batched (copy or link-edit utility only), SMP/E debatches the input and retries the utility for each member separately. If this final attempt fails, the resulting x37 abend is treated as an unacceptable utility return code. In this case, processing continues for SYSMODs containing eligible updates to other libraries, but processing fails for SYSMODs containing unprocessed elements for the out-of-space library (and it fails for any SYSMODs that are dependent on the failed SYSMODs). For guidance on setting up the desired retry processing, see the *SMP/E R8.1 User's Guide*. For more information about OPTIONS entries, see "OPTIONS Entry (Global Zone)" on page 741.

If there is no RETRYDDN list, SMP/E does not try to recover from out-of-space errors, even if YES is specified.

**NO**

indicates that SMP/E should not try to recover from the error.

**REUSE**

indicates that if a module was successfully assembled during previous SMP/E processing, it should not be reassembled. Instead, the existing object module from SMPWRK3 should be reused.

## SELECT

specifies one or more SYSMODs that should be accepted.

### Notes:

1. **SELECT** can also be specified as **S**.
2. To reaccept a SYSMOD, it is not enough to specify that SYSMOD on the **SELECT** operand. You must also specify **RED0**.
3. To process functions containing a **++VER DELETE** statement, you must specify them on the **SELECT** operand.

## SOURCEID

indicates that SYSMODs associated with the specified source IDs should be accepted.

### Notes:

1. There are two ways to specify source IDs:
  - Explicitly, by specifying a particular source ID (for example, **PUT9303**). In this case, only that particular source ID is used.
  - Implicitly, by specifying either **\*** or **c\*** (for example, **PUT\***), where **c** is a 1- to 7-character string. In the second case, all source IDs beginning with the specified character string are used.
2. A given source ID can be explicitly specified **only once** on the **SOURCEID** operand.
3. The same source ID can **not** be explicitly specified on both the **EXSRCID** and **SOURCEID** operands.
4. If a source ID is specified implicitly or explicitly on both the **SOURCEID** operand and the **EXSRCID** operand, all SYSMODs with that source ID are excluded from processing.
5. If a given SYSMOD has multiple source IDs and at least one of those source IDs is specified either explicitly or implicitly on the **SOURCEID** operand, and another one is specified either explicitly or implicitly on the **EXSRCID** operand, the SYSMOD is excluded from processing.  
  
For example, assume PTF UZ12345 has been assigned source IDs **SMCREC** and **PUT9303**. If you specify **SOURCEID(SMC\*)** and **EXSRCID(PUT9303)**, the SYSMOD is excluded from processing.
6. Functions containing a **++VER DELETE** statement are not automatically included by the **SOURCEID** operand. You must specify them on the **SELECT** operand.
7. If no SYSMOD types are specified, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.

## USERMODS

indicates that all eligible USERMODs should be accepted.

### Notes:

1. **USERMODS** can also be specified as **USERMOD**.
2. If **USERMODS** is specified along with **SELECT**, all eligible USERMODs are included, in addition to the SYSMODs specified on **SELECT**.
3. If **USERMODS** is specified along with **SOURCEID**, all USERMODs associated with the specified source IDs are included.

## Syntax Notes

Figure 1 on page 18 shows how SMP/E chooses which SYSMODs to process, on the basis of the operands specified on the **ACCEPT** command.

- If you specify any of the operands in the top part of the chart, or if you do not specify the **SELECT** operand, SMP/E does **mass-mode** processing.
- If you specify the **SELECT** operand, SMP/E does **select-mode** processing.
- If you specify the **SELECT** operand plus operands from the top part of the chart, SMP/E does both **select-mode** and **mass-mode** processing.

For more information about select-mode and mass-mode processing, see “Candidate Selection” on page 29.

Remember the following when coding the **ACCEPT** command:

- SMP/E accepts SYSMODs specified on **SELECT**, regardless of other **ACCEPT** operands (such as a SYSMOD type, **SOURCEID**, **EXSRCID**, or **FORFMID**). Therefore, if you want to accept a specific SYSMOD, you only need to specify the SYSMOD ID on **SELECT**. For example, to accept a specific APAR, you do not also have to include the APAR operand.  
**Note:** If you do specify a SYSMOD type along with **SELECT**, SMP/E accepts all SYSMODs of the specified type plus the selected SYSMOD.
- If you specify more than one SYSMOD type, a SYSMOD needs to match only one of the specified types.
- If the **SOURCEID**, **FORFMID**, and SYSMOD type operands are specified together, only those SYSMODs meeting all the conditions are accepted.

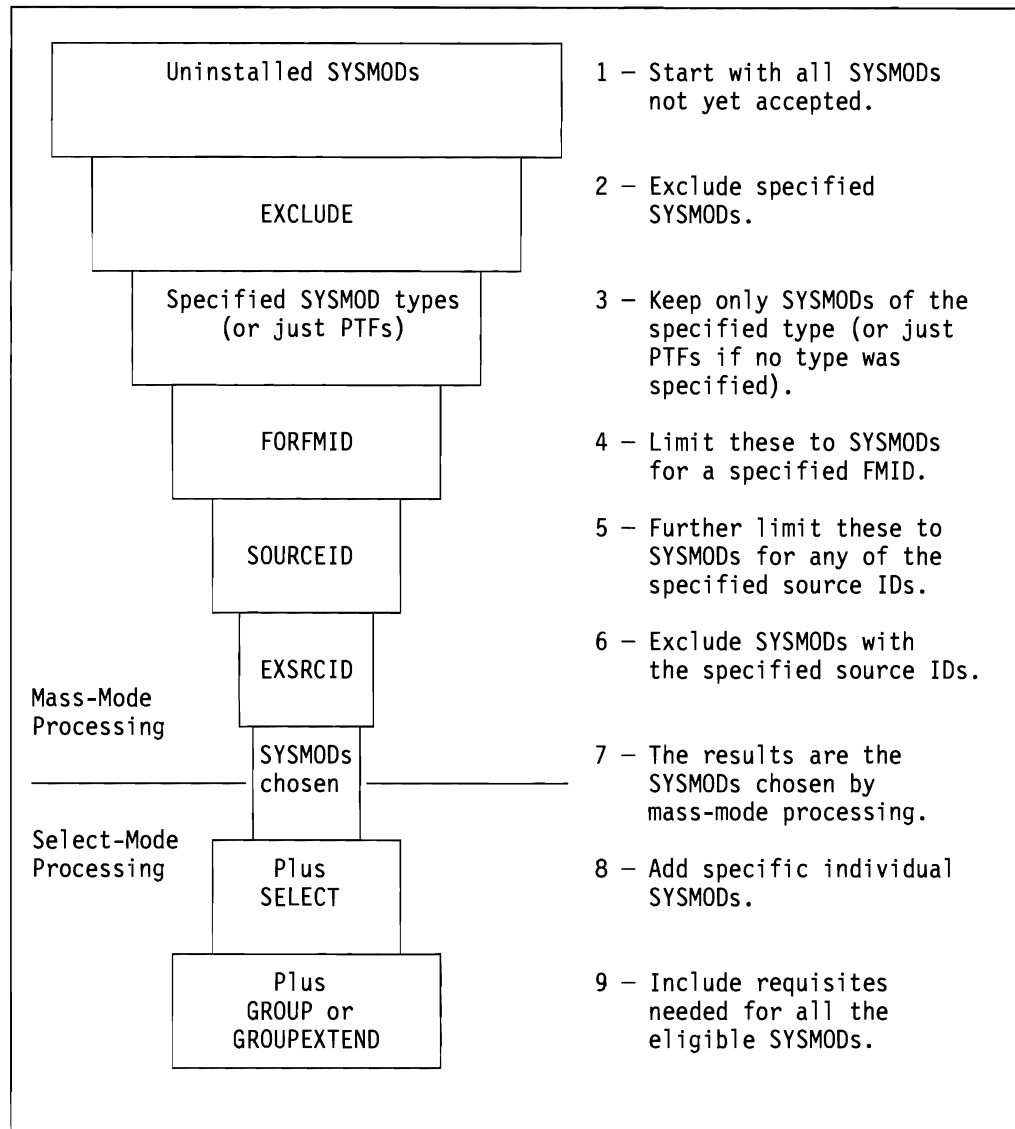


Figure 1. Combining SYSMOD Selection Operands on the ACCEPT Command

## Data Sets Used

The following data sets may be needed to run the ACCEPT command. They can be defined by DD statements or, preferably, by DDDEF entries. For more information about these data sets, see Chapter 34.

PARMLIB	SMPPTS	SMPWRK2	SYSUT2
SMPCNTL	SMPRPT	SMPWRK3	SYSUT3
SMPCSI	SMPSCDS	SMPWRK4	SYSUT4
SMPLOG	SMPSNAP	SMPWRK6	Distribution library
SMPLOGA	SMPSTS	SYSLIB	Link library
SMPMTS	SMPTLIB	SYSPRINT	Text library
SMPOUT	SMPWRK1	SYSUT1	zone

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified,

---

the data sets are dynamically allocated using the ZONEINDEX information in the GLOBALZONE entry.

---

## Usage Notes

This section describes the following:

- “Adding New Elements to the Distribution Libraries”
- “DISTLIB Operand Checking”
- “DISTSRC, ASSEM, and DISTMOD Operands” on page 20
- “Alias Processing” on page 20
- “ACCEPT CHECK Facility” on page 21
- “SYSMOD Termination” on page 21
- “ACCEPT Termination” on page 23
- “Automatic Reinstallation of SYSMODs” on page 23

## Adding New Elements to the Distribution Libraries

Any SYSMOD can introduce a new element to the distribution libraries without any processing outside the scope of SMP/E. To add a new element, you just have to identify the distribution library (that is, the DISTLIB operand on the appropriate MCS). SMP/E assumes the functional level to be that to which the SYSMOD is applicable, and the service level to be from the SYSMOD itself.

## DISTLIB Operand Checking

When an element is selected to be installed and a distribution zone entry for that element already exists, the value of the DISTLIB operand on the element MCS is compared with the DISTLIB subentry in the distribution zone element entry. For service SYSMODs and elements other than modules in dependent functions, if the DISTLIB values are not equal, SMP/E issues a message to inform you of an error condition and terminates the SYSMOD containing the element. For modules in dependent function SYSMODs, the element is moved to the new distribution library and deleted from the old library.

**Note:** DISTLIB checking is not done for base function SYSMODs, because the SYSMOD defines the initial distribution libraries for its elements.

If service and function SYSMODs containing the same element are being processed, and no element entry exists on the distribution zone, the service SYSMODs must specify the same DISTLIB as the function SYSMODs on the element MCSs. If they do not, SMP/E issues an error message and the service SYSMOD is terminated.

If two service SYSMODs update or replace the same element, have different DISTLIB operand values, and are both eligible for processing, but no entry for the element exists on the distribution zone, the first SYSMOD processed causes a distribution zone element entry to be created containing the DISTLIB from its MCS. SMP/E terminates the second SYSMOD.

## DISTSRC, ASSEM, and DISTMOD Operands

Because SMP/E cannot determine from the data processed by JCLIN what source is contained in a totally copied library, the DISTSRC, ASSEM, and DISTMOD operands are provided to pass this information to SMP/E when a macro being replaced or updated, must cause the reassembly of source.

- The DISTSRC operand value specifies the name of the distribution library containing the source.
- The ASSEM and PREFIX operand values specify a list of sources that should be assembled during APPLY processing.
- The DISTMOD operand value specifies the name of the distribution library containing the load modules.

These four operands are specified on ++MAC and ++MACUPD statements. The DISTMOD operand is also specified on ++SRC and ++SRCUPD statements.

The ASSEM operand values are placed in the associated SYSMOD entry on the distribution zone as ASSEM subentries. If any of the modules specified in the ASSEM operand values are found on the target zone as SRC or ASSEM entries, the DISTLIB and SYSLIB subentry values are used in lieu of the DISTSRC operand value.

If neither a SRC nor an ASSEM entry exists for a module in the ASSEM operand values, a SRC entry is created. The DISTSRC operand value is placed in the SRC entry as the DISTLIB subentry.

If there is no MOD entry on the distribution zone for a module in the ASSEM operand list, one is created. The DISTMOD operand value is placed in the MOD entry as the DISTLIB subentry.

After the macro update or replacement is accomplished, all modules specified in the ASSEM and PREFIX operand lists are assembled. If no member is found in the source distribution library or in the distribution library for a source specified in the ASSEM operand list, a warning message is issued, and processing of the SYSMOD continues without assembling or link-editing the module. If an assembly completes with a return code greater than the one you specified in the RC subentry of the ASM UTILITY entry (or the SMP/E default of 4 if the RC subentry is null), the processing of the SYSMOD stops. If the resulting object text from a successful assembly can be link-edited into a load module, the link-edit is performed.

## Alias Processing

When an element with aliases is processed, both the element and its aliases are updated. SMP/E does not check the aliases against elements maintained in the distribution zone. The user must make sure an element's alias does not match the name of an element maintained by SMP/E in the distribution zone.



Aliases for an element are determined as follows:

- Replacement elements (MACs, MODs, and data elements):
  1. If a list of aliases is specified on the SMP/E MCS, these aliases are used. Aliases (in the distribution library) existing before this new list of aliases was presented to SMP/E are not replaced. They remain in the distribution library. Further, the new list replaces any alias subentries in the distribution zone element entry.
  2. If no list of aliases is specified on the SMP/E MCS, the aliases found as alias subentries in the distribution zone element entry are used.
- Update elements (ZAPs and MACUPDs):
  1. If a list of aliases is specified on the SMP/E MCS, these aliases are used. Any alias subentries in the distribution zone element entry are ignored for update processing of the element. Macro aliases (in the distribution library) existing before this list of aliases was presented to SMP/E are not updated. They remain in the distribution library. Alias subentries in the distribution zone element entry are not updated or replaced by the aliases in this list.
  2. If no list of aliases is specified on the SMP/E MCS, the aliases found as alias subentries in the distribution zone element entry are used.

## ACCEPT CHECK Facility

The intent of the CHECK option is to perform a test run informing you of possible error conditions and providing reports of SYSMOD status, libraries that will be updated, regression conditions, and SYSMODs that will be deleted. During CHECK processing, the list of distribution zone entries is maintained in storage; data is written to the distribution zone as a temporary storage medium. CHECK processing deletes any data written to the distribution zone. Consequently, no permanent updates are made to the distribution zone.

## SYSMOD Termination

Termination of a SYSMOD causes a return code of 8. Termination of a ++FUNCTION causes a return code of 12. Termination occurs in response to any of the following conditions:

- Missing requisites:
  - The requisite SYSMOD is not available on the PTS/CSI data sets. (It has not been received.)
  - The requisite SYSMOD has been excluded.
  - The requisite SYSMOD was terminated (possibly because of other missing requisites).
  - The requisite SYSMOD did not meet the applicability criteria.
  - The requisite SYSMOD was not included in the SELECT list, and neither **GROUP** nor **GROUPEXTEND** was specified.
  - **GROUP** was specified to include the requisite, but the requisite SYSMOD is being held or is not available on the PTS or CSI data sets. (It has not been received.)

- **GROUPEXTEND** was specified to supersede the failing requisite, but a superseding SYSMOD was not available for processing.
- MODID error conditions.
- Attempting to change the ownership of an element that is being updated rather than replaced.
- DISTLIB operand checking failure.
- DD statement missing for a distribution library.
- Utility return codes: Return codes from the utilities called to update, assemble, copy, and link-edit elements to the distribution library are examined to determine the success or failure of an operation. If these return codes exceed a predefined value, the SYSMODs whose elements are involved in the operation are terminated. For details on handling x37 abends, see the description of the RETRY operand under “Operands” on page 7.
- Related SYSMOD failure: When SMP/E excludes an element from a SYSMOD because another SYSMOD being processed supplies a higher level of the element, SMP/E does not consider the first SYSMOD successfully processed until the SYSMOD supplying the highest (selected) level element completes successfully. If the SYSMOD supplying the highest level element fails, all SYSMODs from which elements have been excluded are terminated because of a “related SYSMOD failure.”

### Avoiding SYSMOD Termination

**BYPASS:** Certain error conditions that cause the termination of a SYSMOD can be avoided by specifying the BYPASS operand on the ACCEPT command. In BYPASS mode, some error conditions are treated as warning conditions. The following operand values can be specified with the BYPASS operand to avoid termination:

- |              |   |
|--------------|---|
| <b>ID</b>    | Specifies that SYSMODs should be processed even though their MODID verification checks have failed.                 |
| <b>PRE</b>   | Specifies that SYSMODs should be processed even though their PRE requisite conditions are not met.                  |
| <b>REQ</b>   | Specifies that SYSMODs should be processed even though their REQ requisite conditions are not met.                  |
| <b>IFREQ</b> | Specifies that SYSMODs should be processed even though their conditional requisite conditions (IFREQs) are not met. |

**Utility Return Code Thresholds:** The value SMP/E uses to determine the success or failure of a called utility is kept in the UTILITY entries and can be changed by UCLIN.

## ACCEPT Termination

Termination can be caused by any of the following conditions. For each condition, SMP/E issues an error message:

- Termination of processing of any function SYSMOD.
- Two function SYSMODs are specified in the SELECT list and one specifies the other in the DELETE operand of its ++VER statement.
- Two function SYSMODs are specified in the SELECT list, or are selected in mass mode, and one specifies the other in the NPRES operand of its ++VER statement.
- A function SYSMOD specifying a previously accepted SYSMOD in the NPRES operand of its ++VER statement is specified in the SELECT list.
- A function SYSMOD specified in the SELECT list has been deleted by a previously accepted SYSMOD; that is, a SYSMOD entry on the distribution zone indicates that the SYSMOD has been deleted.
- A function SYSMOD specified in the SELECT list has been superseded by a previously accepted SYSMOD; that is, a SYSMOD entry on the distribution zone indicates that the SYSMOD is superseded. A service SYSMOD in the same situation is not processed, but the ACCEPT command is not terminated.
- A function SYSMOD is terminated before selection processing is complete. SMP/E issues a return code of 12 and does not produce a SYSMOD status report.

## Automatic Reinstallation of SYSMODs

The selection of a function SYSMOD that is being accepted for the first time may cause a SYSMOD that was accepted earlier to be selected for reinstallation. This can occur if the modification is applicable to more than one function. For example, consider the following SYSMOD:

```
++PTF(UZ00001) .
++VER(Z038) FMID(GVT3100) .
++IF      FMID(GVT3101) THEN REQ(UZ00001) .
++VER(Z038) FMID(GVT3101) .
++MOD(IFTABCD) DISTLIB(AOS99) .
```

If this PTF was first accepted when only function GVT3100 was installed, the first ++VER statement would have been used and the conditional requisite data supplied on the ++IF would have been saved. If GVT3101 is subsequently installed, the saved ++IF data would require reinstallation of this same PTF.

**Note:** Because SMP/E does not process a SYSMOD with more than one VER that appears to be valid, GVT3101 must DELETE GVT3100 for this construction to work properly.

## Output

The following reports may be produced during ACCEPT processing:

- Causer SYSMOD Summary report
- Deleted SYSMOD report
- File Allocation report
- Element Summary report
- JCLIN Cross-Reference report
- JCLIN Summary report
- MOVE/RENAME/DELETE report
- SYSMOD Regression report
- SYSMOD Status report

These reports are described in Chapter 31.

---

## Examples

The following examples are provided to help you use the ACCEPT command:

- “Example 1: Accepting All SYSMODs from a Given Source”
- “Example 2: Accepting All SYSMODs for Selected Functions” on page 25
- “Example 3: Accepting with the GROUP Operand” on page 25
- “Example 4: Accepting with the GROUPEXTEND Operand” on page 26
- “Example 5: Accepting with the CHECK Operand” on page 26
- “Example 6: Combining ACCEPT Operands” on page 27
- “Example 7: Doing ACCEPT before APPLY” on page 27
- “Example 8: Installing Service for All ESO Service Levels” on page 28
- “Example 9: Excluding SYSMODs with Certain Source IDs” on page 28

### Example 1: Accepting All SYSMODs from a Given Source

If you used the SOURCEID operand during RECEIVE processing to group all the SYSMODs processed, you may choose to install only that set of SYSMODs. You can do this with the SOURCEID operand of the ACCEPT command. Suppose you received an ESO containing service levels PUT9301 and PUT9302. The ESO contained ++ASSIGN statements that assigned each PTF a SOURCEID value corresponding to the service level it is part of. Now you want to install all the applicable PTFs from those tapes into the distribution libraries described by zone MVSDLB1. You can do this with the following commands:

```
SET      BDY(MVSDLB1)      /* Process MVSDLB1 DLIB zone. */.
ACCEPT  SOURCEID(PUT9301, /* Process these service */
          PUT9302)        /* levels */.
          GROUP           /* and any requisites. */.
```

## Example 2: Accepting All SYSMODs for Selected Functions

At times, you may only want to install changes for a single function or for a certain group of functions. You can do this with the FORFMID operand on the ACCEPT command. Assume you want to install service for function JXX1234 and for all the functions on your system that are related to telecommunication. You first need to define an FMIDSET for the telecommunication functions. You can do this with the following commands:

```

SET      BDY(GLOBAL)      /* Process global zone.      */.
UCLIN                                /* UCLIN to set up          */.
                                FMIDSET.                          */.
ADD      FMIDSET(TC)      /* Define TC FMIDSET.        */.
                                FMID(JXX0001                       */.
                                JXX0002)                          */.
                                /*                               */.
ENDUCL   save             /* End UCL set up.          */.

```

You can now use the following commands to install PTFs for function JXX1234 and for the functions in FMIDSET TP:

```

SET      BDY(MVSDLB1)     /* Process MVSDLB1 DLIB zone. */.
ACCEPT   FORFMID(JXX1234, /* ACCEPT for selected FMIDs. */.
                                TC)                               /*                               */.

```

## Example 3: Accepting with the GROUP Operand

At times, you may know that a particular SYSMOD is required on your system, but you may not know all its requisite SYSMODs. You can use the GROUP operand of ACCEPT to have SMP/E determine all the requisites and install them automatically. This method is often used when a new function is being installed. Suppose you want to install a new function, HYY1234, with all its service and any requisite SYSMODs. You can do this with the following commands:

```

SET      BDY(MVSDLB1)     /* Process MVSDLB1 DLIB zone. */.
ACCEPT   FORFMID(HYY1234) /* For one function.          */.
                                FUNCTIONS PTFs                       /* Functions and PTFs        */.
                                GROUP                               /* plus requisites.         */.

```

The FORFMID operand indicates that only SYSMODs applicable to this function should be installed. The FUNCTIONS operand indicates that HYY1234 can be installed. The PTFs operand indicates that only PTFs for HYY1234 should be installed (no APARs or USERMODs are included). The GROUP operand indicates that **all** requisite SYSMODs should also be accepted. These requisites can be applicable to other functions but may not be APARs or USERMODs.

## Example 4: Accepting with the GROUPEXTEND Operand

Assume you want SMP/E to automatically include the requisites for some SYSMODs you plan to install. However, you are not sure whether all of the requisites are available. (They may not have been received, or they might be held because they are in error.) In these cases, you would like SMP/E to check whether a superseding SYSMOD is available for the unsatisfied requisites. To have SMP/E do this additional checking, you can use the GROUPEXTEND operand:

```
SET      BDY(MVSDLB1)      /* Process MVSDLB1 DLIB zone. */.
ACCEPT  FORFMID(HYY1234) /* For one function.          */.
        FUNCTIONS PTFs    /* Functions and PTFs plus   */.
        GROUPEXTEND      /* requisites or supersedes. */.
```

SMP/E accepts HYY1234 and any functions or PTFs applicable to HYY1234. Because of the GROUPEXTEND operand, SMP/E also accepts all requisites for those SYSMODs, even if the requisites are not applicable to HYY1234. If SMP/E cannot find a requisite, it looks for a SYSMOD that supersedes the requisite and uses it to satisfy the requirement. Likewise, if a requisite is being held for an error reason ID, SMP/E looks for a SYSMOD that supersedes the requisite, or that either satisfies or supersedes its error reason ID, and uses it to satisfy the requirement.

## Example 5: Accepting with the CHECK Operand

In Example 3, SMP/E was directed to automatically include SYSMODs needed for the selected function and service. To review which SYSMODs will be included before you actually install them, you can use the CHECK operand of ACCEPT, as shown in the following commands:

```
SET      BDY(MVSDLB1)      /* Process MVSDLB1 DLIB zone. */.
ACCEPT  FORFMID(HYY1234) /* For one FMID.              */.
        FUNCTIONS PTFs    /* Functions and PTFs        */.
        GROUP           /* plus requisites           */.
        CHECK           /* in check mode.            */.
```

After running this command, you should check the SYSMOD Status Report to see which SYSMODs would have been installed if you had not specified CHECK. If the results of this trial run are acceptable, you can run the commands again without the CHECK operand to actually install the SYSMODs.

## Example 6: Combining ACCEPT Operands

You may want to further divide the work to be done by specifying combinations of the ACCEPT operands. The following is an example using all the SYSMOD selection operands of ACCEPT:

```

SET      BDY(MVSDLB1)      /* Process MVSDLB1 DLIB zone. */.
ACCEPT  SOURCEID(PUT9301  /* For these service levels. */
          PUT9302)        /*                               */
          FORFMID(HYY1234 /* For selected functions    */
          TP)             /*                               */
          FUNCTIONS PTFs  /* install all type SYSMODs  */
SELECT  (UZ00001         /* plus these three SYSMODs  */
          UZ00002         /* for other functions,      */
          UZ00003)        /*                               */
EXCLUDE(UZ00010         /* but not these three,     */
          UZ00011         /*                               */
          UZ00012)        /*                               */
GROUP   /* plus all requisites. */.

```

By issuing these commands, you direct SMP/E to accept all the SYSMODs from service levels PUT9301 and PUT9302 that are applicable either to function SYSMOD HYY1234 or to one of the function SYSMODs identified in the FMIDSET entry TP. Any other SYSMODs required to install those SYSMODs are also installed. Both FUNCTIONS and PTFs SYSMODs are eligible for selection. In addition, SYSMODs UZ00001, UZ00002, and UZ00003 are accepted, even though they are not part of PUT9301 or PUT9302, and they may not belong to function SYSMOD HYY1234 or to FMIDSET entry TP. SMP/E does not install SYSMOD UZ00010, UZ00011, or UZ00012, even if they are requisites for other eligible SYSMODs.

## Example 7: Doing ACCEPT before APPLY

Assume you want to install PTFs from PUT9301 and PUT9302 into the distribution libraries to prepare for a full system generation. Therefore, you have not applied the SYSMODs before accepting them. To do this, you must use the BYPASS(APPCHK) operand to have SMP/E ignore whether the PTFs have been applied. You can use the following commands:

```

SET      BDY(MVSDLB1)      /* Process MVSDLB1 DLIB zone. */.
ACCEPT  SOURCEID(PUT9301  /* For these service levels  */
          PUT9302)        /*                               */
          GROUP           /* plus all requisites.     */
          BYPASS(HOLDSYS /* OK to install SYSMODs  */
          (FULLGEN      /* that need SYSGEN or    */
          IOGEN)        /* IOGEN.                  */
          APPCHK)       /* OK to accept before    */
                   /* apply.                   */.

```

## Example 8: Installing Service for All ESO Service Levels

Assume you want to install the preventive service received from all ESO tapes into zone MVSESA1 without having to specify all possible ESO service levels on the SOURCEID operand. You can use the following commands:

```
SET      BDY(MVSESA1)      /* Process MVSESA1 DLIB zone. */.
ACCEPT  PTFS              /* Install all PTFs           */.
        SOURCEID(PUT*)    /* for all service levels    */.
        CHECK             /* in check mode.           */.
```

## Example 9: Excluding SYSMODs with Certain Source IDs

Assume you have received an ESO with PTFs up to service level PUT9303 and you now want to install service from all but the latest two service levels (PUT9302 and PUT9303) into zone MVSESA2. You can use the following commands:

```
SET      BDY(MVSESA2)      /* Process MVSESA2 DLIB zone. */.
ACCEPT  PTFS              /* Install all PTFs           */.
        SOURCEID(PUT*)    /* for all service levels    */.
        EXSRCID(PUT9302   /* except for PUT9302       */.
              PUT9303)    /* and PUT9303,             */.
        GROUPEXTEND      /* and any requisites       */.
        CHECK             /* in check mode.           */.
```

## Processing

Generally, ACCEPT processing is very similar to APPLY processing, except that the distribution zone, rather than the target zone, controls processing and the distribution libraries, rather than the target libraries, are updated.

## SYSMOD Selection

This section outlines the process by which SYSMODs and the elements from the SYSMODs are selected.

### Operands Related to SYSMOD Selection

The following ACCEPT command operands can be used to specify to SMP/E which SYSMODs are to be processed:

```
APARS
BYPASS(APPLYCHECK)
EXCLUDE
EXSRCID
FORFMID
FUNCTIONS
GROUP
GROUPEXTEND
PTFS
SELECT
SOURCEID
USERMODS
```



## Candidate Selection

The SYSMOD selection operands of the ACCEPT command can be specified separately or in combination in order to control the SYSMODs that SMP/E is to process. When specified separately, SMP/E selects only those SYSMODs that meet the one selection criterion specified. When you specify them in combination, SMP/E does the following checking to build the complete candidate list:

1. SMP/E assumes that normal SYSMOD processing is:
  - a. RECEIVE
  - b. APPLY
  - c. ACCEPT

Therefore, before SMP/E accepts a SYSMOD, it checks to make sure you have applied it. SMP/E does this checking by looking at the applicable target zone to see if the entry has been installed there (not by looking at any information in the global zone SYSMOD entry). The applicable target zone is determined from the RELATED field in the distribution zone DZONE entry.

In some circumstances, you may want to accept a SYSMOD before it is applied—for example, when preparing the distribution libraries before doing a full system generation. In this case, you must specify the BYPASS(APPLYCHECK) operand, telling SMP/E not to check the target zone to make sure the SYSMOD has been applied.

**Note:** The BYPASS(APPLYCHECK) function was previously provided by the NOAPPLY operand, which is no longer supported.

2. SMP/E checks the global zone and the specified distribution zone to determine which SYSMODs in the global zone and SMPPTS have not already been accepted into the distribution zone.

SMP/E checks each such SYSMOD to see if it meets the criteria of any additional selection operands.

- a. If you specify the EXCLUDE operand, SMP/E makes sure the SYSMOD was not specified in the exclude list.
- b. If you specify one or more of the SYSMOD-type operands (that is, FUNCTIONS, PTFs, APARS, or USERMODS), SMP/E checks to make sure the SYSMOD type was one of those specified.

If you do not specify a SYSMOD-type operand, the default is for SMP/E to process only PTF SYSMODs.
- c. If you specify the FORFMID operand, SMP/E makes sure that either the FMID value on one of the ++VER statements within the SYSMOD or the SYSMOD ID itself matches either an FMID specified in FORFMID or one of the FMID values contained in a specified FMIDSET.
- d. If you specify the SOURCEID operand, SMP/E makes sure one of the SOURCEIDs of the SYSMOD matches a source ID that you have specified, either explicitly or implicitly, on the SOURCEID operand.
- e. If you specify the EXSRCID operand, SMP/E makes sure none of the SOURCEIDs of the SYSMOD matches a source ID you have specified, either explicitly or implicitly, on the EXSRCID operand.

**Note:** If a given SYSMOD has multiple source IDs and you specify at least one of them implicitly or explicitly on the SOURCEID operand,

and another one explicitly or implicitly on the EXSRCID operand, that SYSMOD is excluded from processing.

Similarly, if you specify a given source ID implicitly or explicitly on the EXSRCID operand and also implicitly or explicitly on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.

Each SYSMOD that satisfies all of the above conditions is considered a candidate for the ACCEPT process. In other words, by specifying the SYSMOD type operands, the FORFMID operand, or the SOURCEID operand in combination, you cause SMP/E to select only SYSMODs that meet all the specified conditions.

3. If you specify the SELECT operand, each SYSMOD specified in the select list is considered a candidate, regardless of its SYSMOD type, FMID value, or SOURCEID value. That is, SELECT has an additive effect on the SYSMOD selection. This is called *select-mode* processing. If you do not specify SELECT, this is called *mass-mode* processing.

**Note:** If SELECT is the only operand you specify, only SYSMODs in the select list are processed.

4. If you specify the GROUP or GROUPEXTEND operand, SMP/E checks each of the candidate SYSMODs to determine whether they have any requisites that must also be accepted. SMP/E automatically includes the following SYSMODs, as long as they are not specified on the EXCLUDE operand or excluded by the EXSRCID operand:

- Any SYSMOD not already accepted and specified as a prerequisite (that is, specified in the ++VER statement PRE operand) of one of the candidate SYSMODs
- Any SYSMOD not already accepted and specified as a corequisite (that is, specified in the ++VER statement REQ operand) of one of the candidate SYSMODs
- Any SYSMOD not already accepted and specified as a conditional requisite (that is, specified in the ++IF statement REQ operand) of one of the candidate SYSMODs
- Any SYSMOD not already accepted and specified as a conditional requisite (CIFREQ subentry) in the distribution zone SYSMOD entry for one of the candidate SYSMODs

If one of these requisites is held or not available and you specified GROUPEXTEND, SMP/E checks the global zone for any SYSMODs that have been received and that either supersede the requisite, or that match or supersede its HOLDERERROR reason ID. The lowest-level SYSMOD found is then used to satisfy the requisite.

- If you specified NOAPARS with GROUPEXTEND, SMP/E does not include APARs that resolve error reason IDs for the held requisites.
- If you specified NOUSERMODS with GROUPEXTEND, SMP/E does not include USERMODs that resolve error reason IDs for the held requisites.

Once a SYSMOD is added to the candidate list, it is eligible to be checked for additional requisites. The FORFMID, SOURCEID, and SYSMOD type operands have no effect on SYSMODs brought in because of the GROUP or GROUPEXTEND operand. The following example accepts all those SYSMODs with a source ID of PUT9301 that are applicable to EBB1102, plus any additional SYSMODs that are required, even though their source ID is not PUT9301 or their FMID is not EBB1102:

```
SET      BDY(DLIB1)          /* Set to DLIB1 zone.      */.
ACCEPT  SOURCEID(PUT9301) /*                               */.
        FORFMID(EBB1102) /*                               */.
        GROUP              /*                               */.
```

### Applicability Checking

Once the ACCEPT candidate list is completed, SMP/E performs additional checking to make sure the selected SYSMODs are applicable to the system.

**General Applicability:** SMP/E makes sure that each SYSMOD meets certain requirements before it is accepted by the system:

- Each SYSMOD must contain **at least one** ++VER statement whose SREL value matches one of the SREL values for that distribution zone and whose FMID value (if present) names a function SYSMOD that has been (or is being) accepted. Function SYSMODs having no ++VER statement FMID operand are applicable if the SREL matches.

Each SYSMOD must have **at most one** ++VER statement whose SREL matches the SREL in the distribution zone entry and whose FMID value exists in the distribution zone (or is being accepted) and has not been superseded.

If a SYSMOD is found that contains multiple applicable ++VER statements, SMP/E cannot accept that SYSMOD because SMP/E cannot determine which ++VER statement to use.

**Note:** If an FMID has been superseded or deleted, SMP/E does not consider it accepted. Therefore, a SYSMOD can contain two ++VER statements that apply to two functions, one of which supersedes the other.

- The SYSMOD must not have already been accepted successfully to the distribution zone.
  - To reaccept a SYSMOD, you must specify the SYSMOD in the SELECT operand and include the REDO operand on the ACCEPT command.
  - SYSMODs partially accepted during an earlier invocation are considered eligible for processing without any special action. These SYSMODs are identified by the ACCEPT ERROR indicator in their distribution zone SYSMOD entry.
- The SYSMOD must not have been partially restored during a previous SMP/E RESTORE attempt. A partially restored SYSMOD can be identified by the RESTORE ERROR indicator in its distribution zone SYSMOD entry.
- The SYSMOD must not have been superseded by an earlier SYSMOD. If a SYSMOD is found to be superseded by another SYSMOD being accepted, no elements are selected from the superseded SYSMOD.
- The SYSMOD must not have been explicitly deleted by a previous SYSMOD.

**Unconditional Requisites (PRE and REQ):** Unconditional requisites are SYSMODs that are required in all functional environments. Each SYSMOD must have all its unconditional requisites satisfied. Unconditional requisites are those specified in the SYSMOD's ++VER statement PRE and REQ operands of the SYSMOD. A requisite is considered satisfied if:

- The requisite SYSMOD is already accepted.
- The requisite SYSMOD is superseded by a SYSMOD already accepted.
- The requisite SYSMOD is being accepted.
- The requisite SYSMOD is superseded by a SYSMOD being accepted.

**Conditional Requisites (IFREQ):** Conditional requisites are SYSMODs required only for a particular functional environment. All conditional requisites of each SYSMOD must be resolved. Conditional requisites are specified on the ++IF statement immediately following the applicable ++VER statement. If the function specified in the FMID operand of the ++IF statement is accepted or is being accepted, each SYSMOD specified in the ++IF statement REQ operand must be satisfied. These requisites are satisfied in the same manner as unconditional requisites.

If the function specified in the FMID operand of the ++IF statement is not already installed, these requisites must be satisfied when the function is later installed. SMP/E, therefore, saves the information from the ++IF statement as CIFREQ sub-entries in the distribution zone SYSMOD entry for that function. When the function is accepted, SMP/E checks the CIFREQ subentries for any requisites of previously accepted SYSMODs and ensures that these requisites are satisfied.

**Negative Requisites (NPRE):** If the NPRE operand is specified on a the ++VER statement for a SYSMOD, the specified SYSMOD ID must not already be installed, must not be installed concurrently, and must not be superseded by a SYSMOD being installed concurrently.

**Note:** The NPRE operand is valid only in function SYSMODs and is used to specify one or more mutually exclusive functions.

**Superseding SYSMODs (SUP):** SMP/E checks to make sure the SYSMOD has not been superseded by another SYSMOD that is already installed or by another SYSMOD being accepted concurrently. If the SYSMOD is superseded, it is not accepted, and the superseding SYSMOD is used instead.

**Note:** If the superseding SYSMOD is not processed because it is held or excluded or because some of its requisites are missing, processing continues as though the SYSMOD did not exist. The SYSMOD that would have been superseded is installed instead.

**Exception SYSMODs (HOLD):** SMP/E makes sure each SYSMOD has no unresolved exception data (that is, HOLD reason IDs) associated with it. Exception data is information specified on the ++HOLD statement. Each ++HOLD statement has a REASON operand specifying a character string that identifies the reason why the SYSMOD has been put into exception status. The following types of exception data are supported by SMP/E:

- HOLDERROR
- HOLDSYSTEM
- HOLDUSER

In addition, the ++HOLD statement may contain a CLASS operand, which specifies an alternative way to resolve a reason ID.

Exception data is considered resolved when one or more of the following are true:

- The reason ID associated with the exception is found as a SYSMOD entry in the distribution zone.
- The reason ID associated with the exception is being accepted concurrently or is being superseded by a SYSMOD being accepted concurrently.
- The applicable BYPASS operand (HOLDERROR, HOLDSYSTEM, HOLDUSER, or HOLDCLASS) was specified.

If **all** the reason IDs associated with a given SYSMOD are not resolved (and an appropriate HOLDCLASS is not specified), SMP/E does not accept that SYSMOD. The SYSMOD is treated as though it had been specifically excluded. Messages are issued showing which reason IDs were not resolved, and the SYSMOD and reason IDs are displayed in the SYSMOD Summary report.

Each category of exception data is resolved differently:

- An **error reason ID** is actually the number of the APAR that caused the SYSMOD to be placed in exception status. As subsequent service is processed, the APAR will probably be superseded by a PTF. When this happens, the error reason ID is resolved and the first PTF is automatically processed. Therefore, it is generally not necessary to use the BYPASS operand to process SYSMODs with error reason IDs.

During any mass installation of SYSMODs, it should be expected that some SYSMODs are not accepted, because unresolved APARs are associated with them. During the installation of preventive service, these SYSMODs should not be investigated further; they will be installed later when a subsequent SYSMOD is produced that supersedes the reason ID that is causing them to be held.

During the installation either of corrective service (that is, installing a PTF or an APAR because of a known problem in the system) or of a new function specifically requiring a SYSMOD, the reason IDs listed should be taken as the first piece of data for research. Research may provide a fix for the problem, in which case the SYSMOD and the fix can be accepted concurrently. If a fix is not available, you can either wait for one, or accept the SYSMOD using the appropriate BYPASS operand.

- A **system reason ID** is a 1- to 7-character string used to identify some action that must be taken before or after a SYSMOD is installed. System reason IDs are not SYSMOD IDs and are not specified in the supersede list of a SYSMOD. Therefore, SMP/E does not automatically release them.

SYSMODs with system reason IDs should be accepted by use of the BYPASS(HOLDSYS(*reason-id*)) operand. If you were to remove the system reason ID by using the ++RELEASE statement, you would then be able to install the SYSMOD, but you would also lose the information about any special processing required in order to accept that SYSMOD on another system.

**Note:** If you have specified the ++HOLD statement with the system reason ID was specified inline (within the held SYSMOD) rather than in the SMPHOLD data set, the **only** way to remove the system reason ID is to specify the BYPASS(HOLDSYS(*reason-id*)) operand.

- A **user reason ID** is defined by the user.

Resolution of user reason IDs depends on why you specified them and the scope of the problem.

- A **hold class** is a 1- to 7-character value identifying an alternative way to resolve another hold type. For example, a PTF may be held for an error reason ID, but IBM has determined that the problem it resolves is more critical than the error reflected by the holding APAR. The ++HOLD statement for this PTF would have an error reason ID of the holding APAR and a class value of ERREL. If you decide to install the PTF, you can specify **(BYPASS HOLDCLASS(ERREL))** and avoid having to resolve the error reason ID.

Class values can be resolved **only** with the BYPASS operand.

SMP/E makes sure no SYSMOD is installed that is dependent on another SYSMOD in hold status.

## SYSMOD Installation

After determining which SYSMODs are to be accepted and which elements should be selected from each SYSMOD, SMP/E begins actually installing these elements by performing the following tasks:

1. Determine the order in which the SYSMODs should be processed.
2. Perform delete processing for any SYSMODs in which the DELETE operand is specified on the ++VER statement.
3. Move specified elements, and delete or rename specified LMOD entries if appropriate.

**Note:** Items 3 and 4 are combined and are done as each SYSMOD is processed.

4. Process any inline JCLIN.

**Note:** Inline JCLIN is processed at ACCEPT time only if ACCJCLIN is set in the DLIBZONE entry.

5. Call system utilities to install the selected elements.
6. Update the applicable distribution zone entries.
7. Produce summary reports identifying all processing done.

The following sections describe each of these tasks in greater detail.

### SYSMOD Processing Order

SMP/E orders the processing of the set of SYSMODs being accepted to ensure proper processing of element selection and source/macro update merges.

The order in which the SYSMODs being accepted are processed is determined from the prerequisite (PRE) data supplied on the ++VER statements of the SYSMODs. SYSMODs named as prerequisites are processed before the SYSMODs that name them.

If no prerequisite order can be determined between SYSMODs, function SYSMODs are processed first, followed by service SYSMODs (PTFs, then APARs, then USERMODs).

## Deleted SYSMODs

A function SYSMOD can delete another function by naming the function to be deleted as an operand of the ++VER DELETE operand. SMP/E deletes the function and all FUNCTIONS, PTFs, APARs, and USERMODs dependent on the deleted function. The functions specifically named in the DELETE operand list are considered *explicitly* deleted SYSMODs; all SYSMODs deleted because of their dependency on the explicitly deleted SYSMODs are termed *implicitly* deleted SYSMODs.

When one function SYSMOD deletes another, SMP/E attempts to remove all information on the deleted SYSMOD from the distribution zone. SMP/E also removes from the distribution libraries all elements that are currently owned by the deleted function SYSMOD. The following processing is done:

1. SMP/E determines whether there are any function SYSMODs in the hierarchy of the function SYSMOD being deleted, and considers those function SYSMODs also eligible for delete processing.
2. SMP/E deletes all SYSMODs that have an FMID value equal to one of the function SYSMODs being deleted.
3. SMP/E identifies all the elements that are currently owned by a function SYSMOD that is to be deleted.
4. SMP/E deletes from the distribution libraries all the elements of the SYSMODs to be deleted. If the elements have been successfully deleted, SMP/E deletes the entries for them from the distribution zone.
5. If the distribution zone contains an LMOD entry for a load module composed entirely of modules that are deleted, the LMOD entry is deleted.
6. If the load module contains modules not being deleted, the LMOD entry is not deleted.

However, for each module deleted from the load module, SMP/E adds a MODDEL subentry for the module to the LMOD entry. MODDEL subentries document the connection between the deleted modules and the load module. If any of these deleted modules are ever reintroduced, an LMOD subentry is added to the MOD entries, and the MODDEL subentries are removed from the LMOD entry.

7. The distribution zone SYSMOD entries for all implicitly deleted SYSMODs are deleted. A distribution zone SYSMOD entry is created for each explicitly deleted SYSMOD. This entry has a DELBY subentry naming the function causing the deletion. The SYSMOD entries for the explicitly deleted SYSMODs prevent the deleted function SYSMODs from being reprocessed by ACCEPT.

**Note:** Some functions may both delete and supersede another function. In this case, the SYSMOD entry contains a SUPBY subentry instead a DELBY subentry. This allows SYSMODs naming the deleted function as a requisite to be installed nevertheless.

The result of this process is that all SYSMODs within the hierarchy of the specified function SYSMOD are deleted.

In the example in Figure 2 on page 36, function SYSMODs HDE1203, HDE1303, and HDE1403, and service SYSMODs UZ00009, UZ00010 and UZ00004 are deleted, because DELETE(HDE1203) is specified on the ++VER statement.

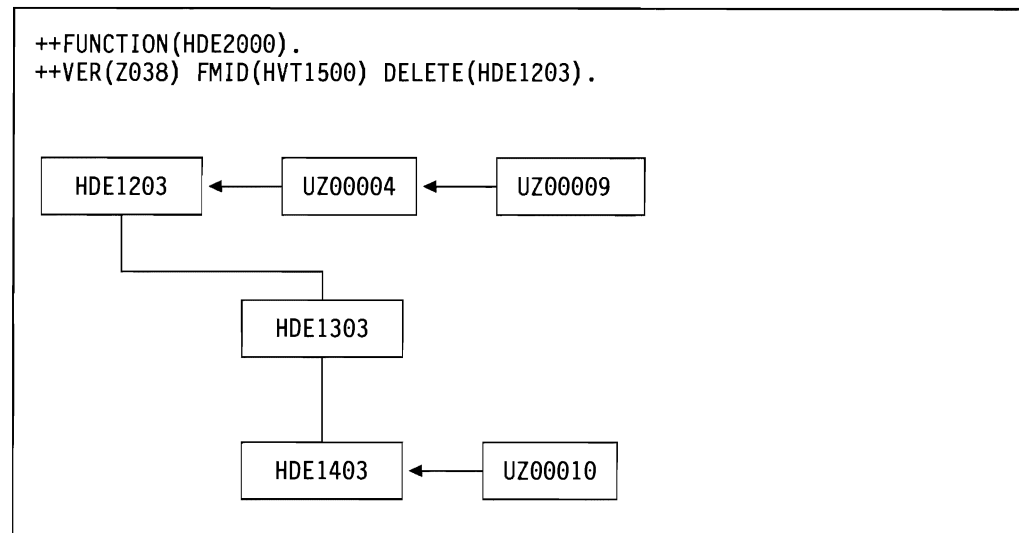


Figure 2. DELETE Hierarchy for DELETE(HDE1203): ACCEPT Processing

CIFREQ subentries in the SYSMOD entry for a function that is deleted (either explicitly or implicitly) are retained in the SYSMOD entry along with the DELBY subentry.

Thus, for the example in Figure 2, when function HDE2000 is applied, CIFREQ subentries in the SYSMOD entry for function HDE1203 are retained, as are any CIFREQ subentries in the SYSMOD entries for functions HDE1303 and HDE1403. Any CIFREQ subentries for conditional requisites specified by the deleted SYSMODs are also retained in the appropriate SYSMOD entries.

**Note:** SMP/E assumes that when a function is deleted, the deleting function replaces all the required elements of the deleted function. Although you can build a function SYSMOD that does nothing but delete another function, it is your responsibility to make sure you still have a functionally complete system after the product has been deleted. One item commonly overlooked is the IHASUxx macros, which are used to indicate whether an SU has been installed. If you delete a product, thus causing its IHASUxx macro to be deleted, and do not replace that macro with your own version, indicating that the SU is not installed, you may lose the system generation capability for that system, because system generation requires that all IHASUxx macros be present.

During ACCEPT processing, when a function is deleted from a distribution zone by another function, its FMID is not removed from the FMID list in the global zone. This is because the deleted function may still be applied in other target zones or accepted in other distribution zones.

### Inline JCLIN

Inline JCLIN can be saved for products without SYSGEN support to make building a new system easier. Inline JCLIN data for a SYSMOD is supplied following the ++JCLIN statement. In order to initialize the distribution zone, JCLIN is processed before elements. Later, when a system is built using the existing distribution libraries and the GENERATE command, this JCLIN data can be copied into the target zone. This eliminates the need for a separate step to obtain JCLIN information for products without SYSGEN support.



Remember the following restrictions when you plan to save inline JCLIN at ACCEPT time:

- You can save inline JCLIN only for products whose distribution libraries were initially built with SMP/E Release 3, or later. You cannot save inline JCLIN for products installed with an earlier release of SMP/E.
- Before using SMP/E Release 3, or later, to build the distribution libraries, you must set the ACCJCLIN indicator in the associated DLIBZONE entry. This tells SMP/E to save inline JCLIN in the distribution zone. If ACCJCLIN was not set when you first built the distribution libraries, it should not be set until the next time the libraries are built.
- After you install a product with SMP/E Release 3, or later, and the ACCJCLIN indicator, you must keep ACCJCLIN set in the DLIBZONE entry. This makes sure that any time you accept service for that product, its JCLIN is updated in the distribution zone.
- The only way to save inline JCLIN in the distribution zone is through the ACCEPT command. The JCLIN command does not update the distribution zone.
- Saving JCLIN at ACCEPT does **not** take the place of a stage 1 SYSGEN for products that **do** have SYSGEN support.
- Because additional data is being saved in the distribution zone, the CSI data set containing the distribution zone may require more DASD space.

**Notes:**

1. Inline JCLIN is not processed for superseded or deleted SYSMODs.
2. Inline JCLIN does **not** cause SMP/E to update the distribution libraries; only the entries in the target and distribution zones are updated. These libraries are updated when SMP/E processes the elements in the SYSMOD. The element statements in the SYSMOD determine which elements should be installed.

The NOJCLIN operand on the ACCEPT command prevents the processing of inline JCLIN. NOJCLIN can be used if the JCLIN contains data that would overlay user-modified entries in the distribution zone.

If you specify NOJCLIN without an operand list, inline JCLIN is not processed for any SYSMOD that was selected and contained a ++JCLIN statement. If you specify NOJCLIN with an operand list, inline JCLIN is not processed for the specified SYSMODs.

For more information about JCLIN processing, see Chapter 8.

### Moving Elements

Macros, modules, and source can be moved from one distribution library to another by use of the ++MOVE statement. This processing is done before element selection.

The ++MOVE statement is further described in Chapter 32.

## Element Selection

SMP/E uses the element statements provided in a SYSMOD to determine which elements should be installed in the distribution libraries. The selection of elements from a SYSMOD is based on relationships among SYSMODs being installed, other SYSMODs already installed, and modification identifiers of the corresponding elements installed in the distribution libraries. Three modification identifiers are kept for each element:

- **FMID (function modification identifier):** The FMID of an element is the function-type SYSMOD that owns the element. Generally, the FMID of an element is established (and later changed) by the installation of a function SYSMOD. The FMID of the element is the function SYSMOD that installed the element in the distribution libraries.
- **RMID (replacement modification identifier):** The RMID of an element is the last SYSMOD that replaced the element (or caused the FMID of that element to change). The RMID of an element is established by the SYSMOD that first introduces the element to the distribution libraries. The RMID of an element is changed by the installation of a SYSMOD that supplies a replacement for the element. An element can be replaced with an element defined by replacement MCSs, or with a module resulting from an assembly.
- **UMID (update modification identifier):** The UMIDs of an element are the set of SYSMODs that have installed updates to the distribution library element. A UMID is added to that set for each SYSMOD that installs an update to the element. Whenever a new replacement for the element is accepted, the set of UMIDs is cleared to start anew with subsequent updates installed for the new replacement. Element updates are ++ZAPs, ++MACUPDs, and ++SRCUPDs.

**Note:** Because data elements can only be replaced and cannot be updated, they do not have UMIDs.

The purpose of element selection is to make sure that the correct functional level of each element is selected and that no service is inadvertently removed from the distribution libraries.

Element selection in SMP/E is divided into three cases:

- The FMID of the SYSMOD being installed matches the FMID of the element in the distribution libraries.
- The FMID of the SYSMOD being installed differs from the FMID of the element on the distribution libraries.
- A function SYSMOD is reinstalled.

The following sections describe processing for each case.

### **FMIDs Match: MODID Verification**

In this case, SMP/E is dealing with elements belonging to the same function, and element processing is based on service relationships expressed by means of the PRE and SUP operands.

The following checks are made for the elements in a SYSMOD to ensure a proper relationship between the SYSMOD being installed and previously installed SYSMODs that supplied the same elements.

**All Elements:** The SYSMOD being installed must specify the RMID of the associated distribution library element on the PRE or SUP operand of its ++VER MCS. If the RMID of the distribution library element is the same as its FMID, the element has not been replaced by any SYSMOD. Therefore, the SYSMOD being installed need not specify the RMID value in the PRE or SUP operand.

If the element being installed is a ++SRC/++SRCUPD or ++MAC/++MACUPD element, resulting in an assembly that replaces a distribution library module, the SYSMOD being installed must specify, as one of its PRE or SUP operand values, the RMID of the corresponding distribution library module to be replaced by the assembly. If the distribution library module is itself the result of an assembly (RMIDASM indicator set in the MOD entry), an exception to this requirement is made, because the reassembly picks up any changes caused by the SYSMOD that last replaced the module through an assembly.

If the SYSMOD being processed does not specify the RMID of the element on the PRE or SUP operand of its ++VER MCS, SMP/E does not accept that SYSMOD, because doing so would regress the service supplied by the SYSMOD represented by the RMID. If you want to allow the RMID SYSMOD to be regressed, the BYPASS(ID) operand of the ACCEPT command can be specified.

**Replacement Elements:** The SYSMOD being installed must be a prerequisite for, or must supersede, all UMIDs associated with the distribution library element.

As above, assemblies resulting from ++SRC/++SRCUPD and ++MAC/++MACUPD elements are considered to be replacement modules; the SYSMOD being installed must specify on the PRE or SUP operand all UMIDs of the corresponding distribution library modules to be replaced by the assembly. No exception is made for a SYSMOD that does not specify, on the PRE or SUP operand, all UMIDs associated with modules that have been assembled, because any UMIDs associated with the module are ZAPs that would be overlaid by a new assembly.

If the SYSMOD being processed does not specify each UMID of the element on the PRE or SUP operand of the ++VER MCS, SMP/E does not accept that SYSMOD, because doing so would regress the service supplied by the SYSMODs that the UMIDs represent. If you want to allow the regression to occur, you can use the BYPASS(ID) operand on the ACCEPT command. The MODID check condition is then reported as a warning, and the elements are installed on the distribution libraries.

**Update Elements:** It is assumed that previous updates are still present and are incorporated with the current update. Therefore, a SYSMOD need not state a relationship (PRE or SUP) to a previous update.

The SYSMOD being installed need not specify each UMID of the element on the PRE or SUP operand of the ++VER MCS. If any element UMIDs in the distribution library are not specified in the SUP or PRE operands, a MODID check warning condition is raised and is reported to the user.

The MODID check warning does not result in the termination of the SYSMOD being installed, and the update is installed on the distribution library. The warning is given because SMP/E is unable to determine with certainty that the two modifications have a relationship or that there is an intersection. Thus, it is the responsibility of the developer or the service team (that is, whoever supplies the update

type SYSMOD) to make sure that this SYSMOD specifies the correct relationships with all previous SYSMODs.

### FMIDs Differ

In this case, SMP/E is dealing with elements belonging to different functions, and element selection is based on functional relationships expressed by means of FMID and VERSION. Elements may be excluded (that is, not selected), and processing of the SYSMOD continues under the assumption that a functionally higher version of the element is already installed on the distribution library.

An element is **excluded** from the SYSMOD being installed unless one of the following conditions is met:

- The function SYSMOD being installed names the FMID of the distribution library element in the ++VER FMID operand. In this case, the function being installed is superior to the function that owns the distribution library element; therefore, the element is selected.
- The MCS associated with the element from the SYSMOD being installed has a VERSION operand, and the FMID of the distribution library element is named in the VERSION list. In this case, the element from the SYSMOD being installed is considered to be functionally superior to the distribution library element, and it is selected.

If there is no VERSION operand on the MCS of an element, the SYSMOD IDs named in the VERSION operand on the ++VER are used as described above.

In this situation, SMP/E may be dealing either with a function SYSMOD or with a nonfunction SYSMOD that is changing the functional ownership (FMID) of the elements.

**Note:** If a SYSMOD containing an element update (++SRCUPD, ++MACUPD, or ++ZAP) attempts to change the ownership (FMID) of the element (via the VERSION operand), the SYSMOD cannot be installed.

When an element is selected, its FMID becomes that of the SYSMOD from which it is selected. No further MODID checking is done for these elements. SYSMODs are constructed so that when the functional ownership of a module changes, either the SYSMOD changing the ownership or data stored in the distribution zone SYSMOD entries (the CIFREQ data) contains sufficient information to prevent any service or functional regressions.

### Reinstalling a Function

Element selection gets more complicated only for **function** SYSMODs that are being reinstalled and have elements that intersect with corresponding elements having the same FMID as themselves (see “FMIDs Match: MODID Verification” on page 38).

The processing for this situation proceeds as in “FMIDs Match: MODID Verification” on page 38. When a MODID check error condition is detected, however, SMP/E checks to determine whether the service level of the distribution library element is higher than that of the element from the SYSMOD being reinstalled. If so, the element from the SYSMOD being reinstalled is not selected, and processing of the SYSMOD continues. If not, the SYSMOD is terminated with a MODID check error.

## ACCEPT CHECK Processing

If you specified the CHECK operand on the ACCEPT command, processing stops at this point. SMP/E produces all the usual ACCEPT reports, assuming that any SYSMOD not already reported as having a problem will be successful during the real ACCEPT run. These reports can be used to determine the following:

- Missing DD statements
- Missing requisite SYSMODs
- Regressions
- SYSMOD in hold exception status
- Processing of inline JCLIN

**Note:** Inline JCLIN is processed at ACCEPT time only if ACCJCLIN is set in the DLIBZONE entry.

## Element Installation

Once the proper SYSMODs have been selected and the proper functional and service level of each element has been determined, SMP/E begins the process of calling utility programs to get the elements installed in the appropriate distribution libraries. The following sections describe the process of installing each of the element types supported by SMP/E.

### Compressing the Distribution Libraries

To have SMP/E compress the distribution libraries before installing SYSMODs, you can use the COMPRESS operand on the ACCEPT command. There are two ways to specify which libraries are to be compressed:

- List the specific libraries on the COMPRESS operand (including libraries that may not be affected by the ACCEPT command).
- Specify ALL on the COMPRESS operand; then only libraries in which elements will be installed by this ACCEPT command are eligible for compression.

Once the libraries have been determined, actual processing depends on the library type:

- For **source** libraries, any source that is to be replaced (not updated) by one of the SYSMODs being installed is deleted from the library.
- For **macro** libraries, no deleting is done because the SYSMOD replacing the macro might fail, and other SYSMODs might cause assemblies that use the macro.
- For **data element and HFS element** libraries, any data element or HFS element that is to be replaced by one of the SYSMODs being installed is deleted from the library.
- For **load** libraries, SMP/E determines which load modules within the library contain only modules being replaced by the SYSMODs being installed. Such load modules are deleted from the library. If a load module contains a module not being replaced, that load module is not deleted.

SMP/E then calls the copy utility to do the actual compress operation.

**Note:** The purpose of deleting the members before compressing the library is to reclaim as much space as possible before attempting to install the

SYSMODs. SMP/E processes one library at a time, first deleting members, and then compressing it.

### **Macro Replacements**

One of the steps in actually updating the distribution libraries is to install macro replacements. Multiple SYSMODs can be accepted, each of which may contain a replacement for the same macro.

When two or more SYSMODs replacing the same macro are accepted concurrently, SMP/E determines the version at the highest function and service level. For a full explanation of how SMP/E determines the functional and service level of an element, see “Element Selection” on page 38.

SMP/E then schedules the actual update of the distribution macro library. The library to be updated is determined from the DISTLIB information in the distribution zone MAC entry. (For further information about the initial setting of the MAC DISTLIB, see “Usage Notes” on page 19.) The actual update is done by calling either the copy utility or the update utility.

- The copy utility is used in these cases:
  - The macro was packaged in a relative file (the RELFILE operand was specified).
  - The macro was packaged in a text library (the TXLIB operand was specified) and it had no alias names (that is, no MALIAS names are in the distribution zone MAC entry and no MALIAS operands are on the ++MAC statement).
  - The macro was packaged inline, it had no alias name, and the SSI operand was not specified.

The SSI operand is ignored when **TXLIB** or **RELFILE** is specified.

- The update utility is called in these cases:
  - The macro was packaged in a text library, and the element had an alias name.
  - The macro was packaged inline and (1) it had an alias name, or (2) the SSI operand was specified.

Upon return from either utility, SMP/E issues a message indicating whether the macro has been replaced successfully.

### **Macro Updates**

After all the macro replacements have been done, any macro updates present can be scheduled. Again, multiple SYSMODs may contain updates for the same macro. When two or more updates to the same macro are being processed concurrently, the text from each is merged. Looking at the sequence numbers in columns 73 to 80, SMP/E processes the updates in the order expressed by the PRE operands on the ++VER statements, by internal defaults, or both, as follows:

- If SMP/E finds a processing order relationship between all of the SYSMODs being processed, the merge occurs according to that order.
- If any of the SYSMODs being processed do not have a processing order relationship with other SYSMODs that do have a processing order, the updates

from the unrelated SYSMODs are merged after the updates from SYSMODs that do have a processing order relationship.

- If SMP/E cannot determine the processing order of the SYSMODs, it merges the updates by SYSMOD type: PTFs first, APARs second, and USERMODs third. Within each type, there is no specified order.

SMP/E then calls the update utility to update the distribution library.

The library to be updated is determined from the DISTLIB information in the MAC entry for the distribution zone. For further information about the initial setting of the MAC DISTLIB, see “Usage Notes” on page 19. Upon return from the update utility, SMP/E issues a message telling whether the update was successful.

### Source Replacements

Another step in actually updating the distribution libraries is to install source replacements. Multiple SYSMODs can be accepted, each of which can contain a replacement for the same source.

When two or more SYSMODs replacing the same source are accepted concurrently, SMP/E determines which version is at the highest function and service level. For a full explanation of how SMP/E does this, see “Element Selection” on page 38.

SMP/E then schedules the actual update of the distribution source library. The library to be updated is determined from the DISTLIB information in the distribution zone SRC entry. (For further information about the initial setting of the SRC DISTLIB, see “Usage Notes” on page 19.) The actual update is done by calling either the update utility or the copy utility.

- The copy utility is used if the source replacement was packaged, either in a text library (that is, the TXLIB operand was specified) or in a RELFILE (that is, the RELFILE operand was specified). The SSI operand is ignored when TXLIB or RELFILE is specified.
- The update utility is called if the source replacement was packaged inline and the SSI operand was specified.

Upon return from either utility, SMP/E issues a message telling whether the source was replaced successfully.

### Source Updates

After all the source replacements have been done, any source updates present can be scheduled. Again, multiple SYSMODs can contain updates for the same source. When two or more updates to the same source are processed concurrently, the text from each is merged. Looking at the sequence numbers in columns 73 to 80, SMP/E processes the updates in the order expressed by the PRE operands on the ++VER statements, by internal defaults, or both, as follows:

- If SMP/E finds a processing order relationship between all of the SYSMODs being processed, the merge occurs according to that order.
- If any of the SYSMODs being processed do not have a processing order relationship with other SYSMODs that do have a processing order, the updates from the unrelated SYSMODs are merged after the updates from SYSMODs that do have a processing order relationship.

- If SMP/E cannot determine the processing order of the SYSMODs, it merges the updates by SYSMOD type: PTFs first, APARs second, and USERMODs third. Within each type, there is no specified order.

SMP/E then calls the update utility to update the distribution library. The library to be updated is determined from the DISTLIB information in the distribution zone SRC entry. (For further information about the initial setting of the SRC DISTLIB, see “Usage Notes” on page 19.) Upon return from the update utility, SMP/E issues a message telling whether the update was successful.

### Assemblies

This section describes the following:

- Assembling source
- Assemblies caused by macros
- Reusing previous assemblies

**Assembling Source:** A SYSMOD may supply both the source (++SRC/++SRCUPD) and an object deck (++MOD) for an element. SMP/E determines whether the object deck can simply be link-edited into the distribution library or whether the source must be assembled. This determination is made by considering the following questions:

- Was **ASSEM** specified on the ACCEPT command?
- Are there any updates to the source that the SYSMOD supplying the object does not know about (UMIDs in the distribution zone SRC entry)?
- Has another SYSMOD assembled the module without this SYSMOD knowing about it (RMID of the distribution zone MOD entry for the module to be assembled)?
- Is the **ASSEMBLE** indicator set for the corresponding MOD?

If the answer to any of the above questions is yes, the module is assembled if SMP/E can determine where the resultant assembled object module should go.

SMP/E knows where to put the assembled object module if there is a distribution zone MOD entry and a resultant object module. That is, the DISTLIB is not SYSPUNCH. (For more information on how SMP/E processes the object module after assembly, see “Module Replacements” on page 45.)

**Assemblies Caused by Macros:** A SYSMOD may supply macros requiring the assemblies of modules. The required assemblies are found in the **ASSEM** and **PREFIX** operands on the ++MAC or ++MACUPD statement. The SRC entry matching the name of the module is used as the source for the assembly. If no SRC entry can be found, no assembly is done.

The SYSMOD may also supply an object deck for the modules to be assembled. To determine whether to do the assembly or to use the object decks supplied in the SYSMOD, SMP/E considers the following questions:

- Was **ASSEM** specified on the ACCEPT command?
- Are there any updates to the macro that the SYSMOD supplying the object does not know about (UMIDs in the distribution zone macro entry)?



- Has another SYSMOD assembled the module without this SYSMOD knowing about it (RMID of the distribution zone MOD entry for the module to be assembled)?
- Is the ASSEMBLE indicator set for the corresponding MOD?

If the answer to any of the above questions is yes, the module will be assembled if SMP/E can determine where the resultant assembled object module should go. SMP/E knows where to put the assembled object module if there is a distribution zone MOD entry and a resultant object module, that is, if the DISTLIB is not SYSPUNCH. (For more information on how SMP/E processes the object module after assembly, see “Module Replacements.”)

Whenever a macro modification in a APAR- or USERMOD-type SYSMOD causes a module to be assembled, the ASSEMBLE indicator in the module's distribution zone MOD entry is set on. If any subsequent PTF-, APAR-, or USERMOD-type SYSMODs contain modifications to macro or source elements affecting a module whose ASSEMBLE indicator has been set, they cause the module to be reassembled in spite of the presence of an object module in the SYSMOD. The reassembly prevents regression of the assembled module during the installation of subsequent SYSMODs that might replace the affected module but do not contain the macro modification introduced by the earlier SYSMOD.

To prevent future reassemblies of modules in which the ASSEMBLE indicator has been set, use the UCLIN function to set it off (DEL).

**Reusing Previous Assemblies:** If SMP/E is run after a failure, assemblies are rerun to ensure that the proper source and macros are used. If the same set of SYSMODs is being processed after a failure, the assemblies run before the failure need not be rerun. If the REUSE operand is specified in the ACCEPT command, the previously assembled objects for the failed SYSMOD are used.

Assembled object decks are stored on the SMPWRK3 data set and remain there until the SYSMODs causing the assemblies are successfully processed. To be able to reuse assemblies, SMPWRK3 must not be scratched after an ACCEPT step.

**Note:** SMP/E does not check to make sure the same SYSMODs are being rerun after a failure; the user must be careful when taking advantage of the REUSE facility.

### Module Replacements

The modules (++MODs and assemblies from ++SRCs) selected from a SYSMOD are link-edited directly into a distribution library. From the appropriate distribution zone MOD entry, SMP/E determines:

- The distribution library (based on the DISTLIB subentry)
- The link-edit characteristics (based on the LEPARM subentry)

Because there is a one-for-one correspondence between the object module being replaced and the members in the distribution library, there is no need to save any link-edit control cards in the distribution zone. When SMP/E link-edits a module into the distribution library, no link-edit include card is generated for the version of the module already there. Therefore, when you replace a distribution library module, you must provide SMP/E with the replacement pieces for all the parts of

that module; that is, if the module contains multiple CSECTs, the SYSMOD must contain all the CSECTs, or the missing ones will be lost.

If the module is packaged in a link library (LKLIB) or a relative file (RELFILE), SMP/E copies the module to the distribution library. Any aliases specified for such a module must exist in the LKLIB or relative file in order to be copied.

**Load Module Attributes and Link-Edit Parameters:** The parameters passed to the link-edit utility include load module attributes such as RENT, REUS, and REFR. SMP/E determines the attributes and parameters to be passed, as follows:

- If SMP/E has determined that the binder is available on the system, the STORENX link-edit parameter is passed to the link-edit utility.
- If LEPARM is specified on the ++MOD statement, the attributes supplied are used, and the corresponding distribution zone MOD entry is updated (or created) with these attributes.
- If LEPARM is not specified and there is a distribution zone MOD entry in which the link-edit attributes are stored, the attributes from the MOD entry are used.
- If LEPARM is not specified and a distribution zone MOD entry does not exist (or exists without link-edit attribute indicators), SMP/E accesses the distribution library to determine the link-edit attributes for the module. If the module is not found on the distribution library, the RENT, REUS, and REFR attributes are used.

The parameters passed to the link-edit utility are the attributes determined above **plus** the link-edit utility parameters specified in the UTILITY entry for link-edit processing.

**Note:** SMP/E provides for special processing for any module with a distribution library of SYSPUNCH. This indicates to SMP/E that the module was assembled during the system generation process and that the assembly was stored in a temporary data set, the SYSPUNCH data set. After being used during the system generation process, that data set is deleted. Therefore, during accept, when SMP/E sees any module going to SYSPUNCH, no processing is done for that module.

### Module Updates

For each ++ZAP element within a SYSMOD, SMP/E determines the distribution library for the module and then attempts to install the superzap to that library.

When installing the ZAP, SMP/E performs two passes: the first to process the VER control cards, and the second to process the REP control cards. The REP pass is performed only if the VER pass for the module is successful.

### Replacements for Data Elements and HFS Elements

Still another step in updating the distribution libraries is the installation of replacements for data elements and HFS elements. Multiple SYSMODs can be accepted, each of which may contain a replacement for the same data element or HFS element. When two or more SYSMODs replacing the element are accepted concurrently, SMP/E determines which version is at the highest function and service level. For a full explanation of how SMP/E determines the functional and service level of an element, see "Element Selection" on page 38.

SMP/E then schedules the actual update of the distribution library. The library to be updated is determined from the DISTLIB information in the distribution zone element entry. The actual update is done by either the copy utility or SMP/E.

- The copy utility is used in these cases:
  - The element was packaged in a relative file (the RELFILE operand was specified).
  - The element was packaged in a text library (the TXLIB operand was specified).
  - The element was packaged inline and was not transformed by GIMDTS.
- SMP/E updates the library if the data element or HFS element was packaged inline and had been transformed by GIMDTS. All control information is removed, the transformed data is changed back to its original format, and the distribution library is updated with the element.

At the end of processing, SMP/E issues a message indicating whether the data element or HFS element has been replaced successfully.

**Note:** The HFS copy utility is not required during ACCEPT processing.

## Recording After Completion

Results of processing are recorded in the following entries:

- “Distribution Zone Element Entries”
- “SMPSCDS BACKUP Entries, SMPMTS MTSMAC Entries, and SMPSTS STSSRC Entries” on page 48
- “Distribution Zone SYSMOD Entries” on page 49
- “Global Zone SYSMOD Entries” on page 50

### Distribution Zone Element Entries

ACCEPT processing creates, modifies, and may delete distribution zone element entries.

- Entry update indicator: When an entry is added by a SYSMOD being processed, the SYSMOD's SYSMOD ID is placed in the LASTUPD subentry of the distribution zone element entry.
- ALIAS subentries: The updates to an element's ALIAS subentries are discussed under “Alias Processing” on page 20.
- LMOD subentries: When the LMOD operand is specified on a ++MOD statement, the values in the operand list are added to the distribution zone MOD entry as LMOD subentries.
- MODID subentries:
  - The FMID subentry is replaced with the FMID of the SYSMOD from which the modification to the element was selected. If the SYSMOD is a function SYSMOD, the FMID is set to the SYSMOD ID of the function itself.
  - The RMID subentry is changed when a replacement element or assembly is accepted. The RMID is set to the SYSMOD ID of the SYSMOD supplying the element.

If a MOD entry is being updated as the result of an assembly for a macro or source, the RMID is replaced with the SYSMOD ID of the SYSMOD supplying the ++MAC or ++SRC, and the RMIDASM indicator is set to reflect this occurrence. The RMIDASM indicator for the module is set even if the actual assembly was suppressed because the SYSMOD supplied an assembled version of the module. (See “Source Replacements” on page 43 and “Assemblies” on page 44 for further information.)

For function SYSMODs, if the replacement element's MCS specified an RMID for the element (the RMID operand), the specified value is used.

- All UMID subentries are deleted when a replacement element is accepted: For function SYSMODs, if the replacement element's MCS specified a list of UMIDs for the element (the UMID operand), these UMIDs replace any existing UMIDs for the element.
- UMID subentries are added when updates for the element are accepted. The UMIDs are the IDs of the SYSMODs supplying the updates. If a SYSMOD with an update modification to an element supersedes another SYSMOD with an update modification to the same element, the UMID subentry for the superseded SYSMOD is deleted from the element entry.

- CSECT names

The CSECT information from the ++MOD statement is saved in the MOD entry for the distribution zone. The information saved is determined in the following manner:

- If the SYSMOD that the selected version of the module came from contained the CSECT operand, the CSECT names there are either added to the distribution zone MOD entry (if no CSECT information was already there) or are used to replace the existing list of CSECT names in the distribution zone MOD entry.
- If the SYSMOD that the selected version of the module came from did not contain the CSECT operand, SMP/E determines whether any other SYSMOD applicable to the same FMID was also being accepted. If so, and if any of those SYSMODs contained the CSECT operand, the CSECT information from the SYSMOD at the highest service level is used.

### **SMPSCDS BACKUP Entries, SMPMTS MTSMAC Entries, and SMPSTS STSSRC Entries**

For each SYSMOD successfully accepted, SMP/E deletes the following entries (if applicable):

- BACKUP entries from the SMPSCDS
- MTSMAC entries from the SMPMTS
- STSSRC entries from the SMPSTS

There is only one exception: If the BYPASS(APPLYCHECK) operand was specified, SMP/E assumes that the SYSMODs have not been applied and, therefore, there are no entries to delete.

**Note:** The correct SMPSCDS, SMPMTS, or SMPSTS data set to use is determined solely by a DD statement in the JCL used when SMP/E is invoked or by a DDDEF entry set up in the distribution zone. In both cases, the data set should be the one used for the related target zone named in the DZONE entry for the distribution zone.

## Distribution Zone SYSMOD Entries

For each SYSMOD processed, a SYSMOD entry is created in the distribution zone. If a SYSMOD entry existed previously (as in the case of reapplication of the SYSMOD), the previous entry is replaced. The entry includes data from the applicable ++VER statement, subentries for each of the elements included in the SYSMOD package, and indicators that are set when ++IF and ++JCLIN statements are present.

A SYSMOD is considered successfully processed when all of its selected elements have been accepted by the appropriate distribution libraries **and** all of its requisites have been successfully processed. Because SMP/E processes any number of SYSMODs with elements in common, some SYSMODs may have elements that need not be installed in a distribution library; such SYSMODs are not considered successfully processed until the SYSMODs supplying the higher level versions of the corresponding elements are successful. If the SYSMOD is not successfully processed, an ERROR status indicator is set in the entry.

SMP/E sets the REGEN indicator in the distribution zone when a SYSMOD is accepted. It does not set REGEN in the target zone when the SYSMOD is applied. However, if a distribution zone is copied into a target zone (for example, as part of a system generation), the REGEN indicator is copied into the target zone. Therefore, REGEN can help you determine how a SYSMOD was installed in the target libraries.

- If REGEN is set in the target zone SYSMOD entry, the distribution zone was copied into the target zone. The SYSMOD was, therefore, installed by use of a generation procedure, such as SYSGEN.
- If REGEN is not set in the target zone SYSMOD entry, the distribution zone was not copied into the target zone. The SYSMOD was, therefore, installed by use of the APPLY command.

**Superseded SYSMODs:** When one SYSMOD is superseded by another, SMP/E makes a record of this by adding the name of the superseding SYSMOD to the entry for the superseded SYSMOD.

- When there is only one superseding SYSMOD, its name is saved in the LASTSUP subentry.
- When there are several superseding SYSMODs, the name of the last one is saved in the LASTSUP subentry, and a complete list is saved in the SUPBY subentry.

If the superseded SYSMOD has not been previously accepted, its distribution zone SYSMOD entry contains only the LASTSUP information. Such a SYSMOD entry is called a “dummy entry.” Because the superseded SYSMOD had not been previously accepted, SMP/E does not know what type of SYSMOD it was (function, PTF, APAR, or USERMOD).

**Deleted SYSMODs:** When one SYSMOD is deleted by another, SMP/E makes a record of this by adding the name of the deleting SYSMOD to the DELBY subentry of the deleted SYSMOD. If the deleted SYSMOD has not been previously accepted, its distribution zone SYSMOD entry contains only the DELBY information. Such a SYSMOD entry is called a “dummy entry.” Although the deleted SYSMOD

had not been previously accepted, SMP/E assumes it was a function SYSMOD, because only function SYSMODs can be explicitly deleted.

**Conditional Requisite Data:** For each SYSMOD named as an FMID in a ++IF statement, a SYSMOD entry is created with CIFREQ subentries representing the conditional requisite requirements.

If the SYSMOD existed previously, the CIFREQ data is simply added to the existing entry; otherwise, a new SYSMOD entry is created to save the CIFREQ data for use if the FMID is installed later.

## Global Zone SYSMOD Entries

For each SYSMOD that is successfully accepted, SMP/E deletes the SMPPTS MCS entry, the global zone SYSMOD entry (with any associated HOLDDATA), and the SMPTLIB data sets for any elements that were packaged in relative files.

There are two exceptions:

- If you specified the BYPASS(APPLYCHECK) operand, the entries and SMPTLIB data sets are not deleted. In this case, SMP/E assumes that the SYSMODs have not been applied and, therefore, may be applied later.
- If you set the NOPURGE indicator in the OPTIONS entry you are using, the entries and SMPTLIB data sets are not deleted. In this case, SMP/E does not delete the entries, because you have instructed it not to.

**Note:** The global zone SYSMOD entry is also deleted when the SYSMOD is rejected.

If the global zone SYSMOD entry is not deleted, the distribution zone name is added to it as an ACCID subentry. Therefore, the ACCID subentries in the global zone reflect the distribution libraries into which each SYSMOD has been accepted.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of ACCEPT processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

### 2. SYSMOD selection

Global zone	—	Read with shared enqueue.
SMPPTS	—	Read with shared enqueue.
Target zone	—	Read with shared enqueue.
DLIB zone	—	Update with exclusive enqueue.

### 3. Element selection

SMPPTS	—	Read with shared enqueue.
DLIB zone	—	Update with exclusive enqueue.

## 4. Utility calling

DLIB zone — Update with exclusive enqueue.

## 5. Global zone update

Global zone — Update with exclusive enqueue.

SMPPTS — Update with exclusive enqueue.

DLIB zone — Update with exclusive enqueue.

## 6. Termination

All resources are freed.





---

## Chapter 3. The APPLY Command

The APPLY command is used to cause SMP/E to install the elements supplied by a SYSMOD into the operating (or target) system libraries. The APPLY process:

- Selects SYSMODs present in the global zone and applicable to the specified target system
- Makes sure all other required SYSMODs have either been applied or are being applied concurrently
- Selects the elements from those SYSMODs on the basis of the functional and service level of those elements in the target system and the relationship between the SYSMODs being installed, making sure that the installation of another SYSMOD does not cause any current service to regress
- Calls system utilities to install the elements into the target system libraries
- Records the functional and service levels of the new elements in the target zone
- Records the application of the SYSMOD in the target zone
- Performs cross-zone processing
- Updates the SYSMOD entries in the global zone

The APPLY process is controlled by:

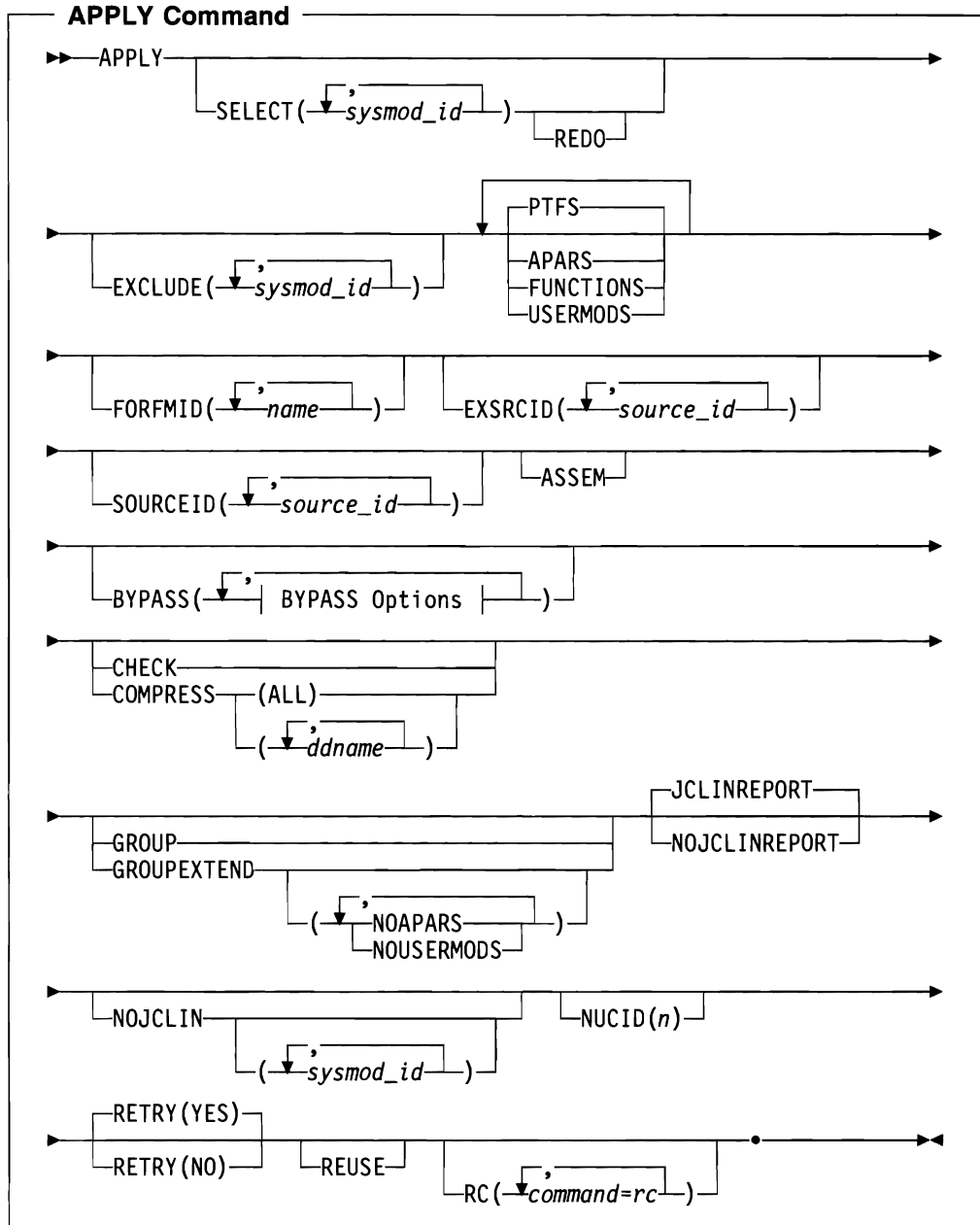
- The information in the target zone reflecting the status and structure of the target system libraries
- Information on the SYSMODs indicating their applicability
- Information in the OPTIONS and UTILITY entries
- Operands on the user's APPLY command

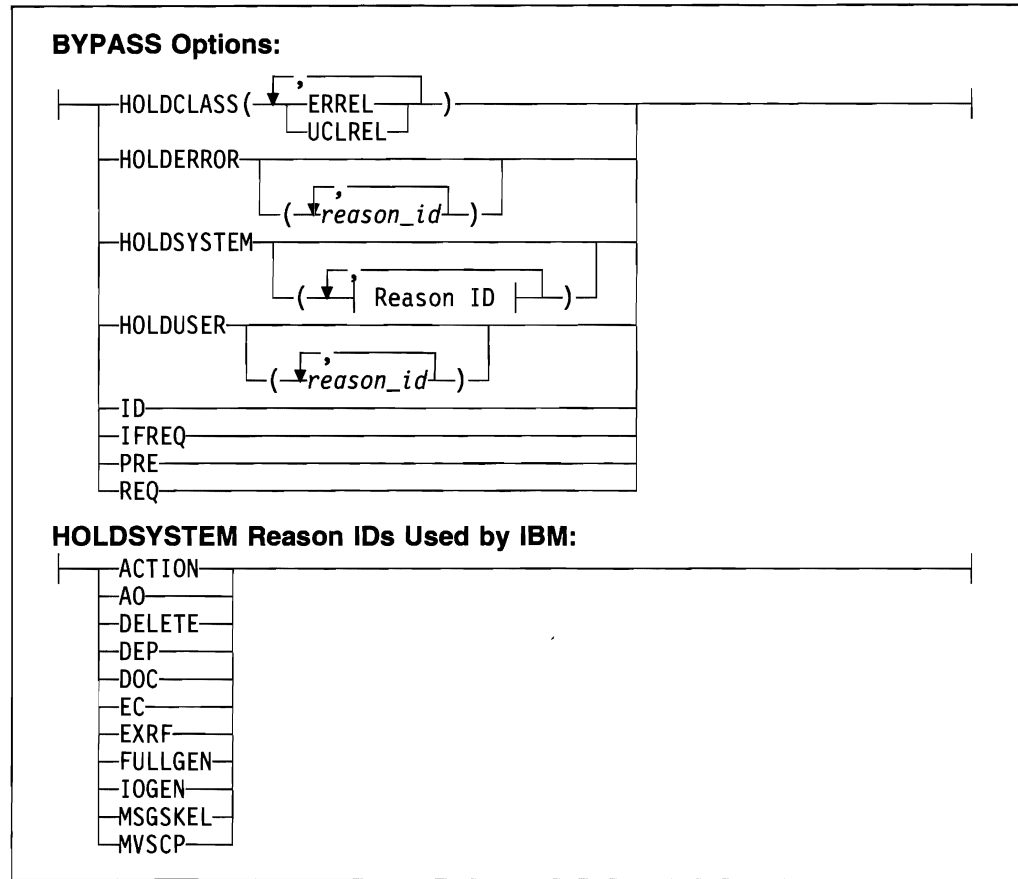
---

### Zones for SET BOUNDARY

For the APPLY command, the SET BOUNDARY command must specify the target zone associated with the target libraries in which the SYSMODs are installed.

Syntax





## Operands

### APARS

indicates that all eligible APARs should be applied.

#### Notes:

1. APARS can also be specified as APAR.
2. If APARS is specified along with SELECT, all eligible APARs are included in addition to the SYSMODs specified on SELECT.
3. If APARS is specified along with SOURCEID, all APARs associated with the specified source IDs are included.

### ASSEM

indicates that if any SYSMOD contains both source code and object code for the same module, the source code should be assembled and should replace the object code.

## BYPASS

You can specify any of these options:

HOLDCLASS  
HOLDERROR  
HOLDSYSTEM  
HOLDUSER  
ID  
IFREQ  
PRE  
REQ

**Note:** If you specify both **BYPASS** and **GROUPEXTEND**, SMP/E does not include superseding SYSMODs needed to take the place of requisites that have been bypassed.

During **CHECK** processing, if you want to see whether any superseding SYSMODs are available for requisites that have been bypassed, specify **GROUPEXTEND** without **BYPASS**.

### **BYPASS(HOLDCLASS(value,...))**

indicates that exception SYSMODs associated with the specified class names should not be held. The list of class names is required.

These are the hold classes you can specify:

<b>Class</b>	<b>Explanation</b>
--------------	--------------------

<b>ERREL</b>	The SYSMOD is held for an error reason ID but should be installed anyway. IBM has determined that the problem the SYSMOD resolves is significantly more critical than the error reflected by the holding APAR.
--------------	--

<b>UCLREL</b>	UCLIN needed for the SYSMOD has been handled by IBM and no longer requires your attention.
---------------	--

### **BYPASS(HOLDERROR)**

indicates that exception SYSMODs associated with the specified error reason IDs should not be held. The list of reason IDs is optional.

If you include a list of reason IDs, only the ones you specify are bypassed. If you do not include a list, all error reason IDs are bypassed.

**Note:** **HOLDERROR** can also be specified as **HOLDERR**.

### **BYPASS(HOLDSYSTEM)**

indicates that exception SYSMODs associated with the specified system reason IDs should not be held. The list of reason IDs is optional. Generally, you should specify **BYPASS(HOLDSYSTEM)** on all **APPLY CHECK** commands, and **BYPASS(HOLDSYSTEM(reason\_id,...))** on all **APPLY** commands for all system reason IDs for which appropriate action has been (or will be) taken.

If you include a list of reason IDs, all the SYSMODs with the specified reason IDs are bypassed. You should, therefore, make sure the appropriate action has been taken for all these SYSMODs. If you do not include a list, all system reason IDs are bypassed.

**Note:** **HOLDSYSTEM** can also be specified as **HOLDSYS**.

These are the system reason IDs currently used by IBM:

**ID Explanation**

**ACTION** The SYSMOD needs special handling before or during APPLY processing, ACCEPT processing, or both.

**AO** The SYSMOD may require action to change automated operations procedures and associated data sets and user exits in products or in customer applications. The PTF cover letter describes any changes (such as to operator message text, operator command syntax, or expected actions for operator messages and commands) that can affect automation routines.

**DELETE** The SYSMOD contains a ++DELETE MCS, which deletes a load module from the system.

**DEP** The SYSMOD has a software dependency.

**DOC** The SYSMOD has a documentation change that should be read before the SYSMOD is installed.

**EC** The SYSMOD needs a related engineering change.

**EXRF** The SYSMOD must be installed in both the active and the alternative MVS/XA\* Extended Recovery Facility (XRF) systems at the same time to maintain system compatibility. (If you are not running XRF, you should bypass this reason ID.)

**FULLGEN** The SYSMOD needs a complete system or subsystem generation to take effect.

**IOGEN** The SYSMOD needs a system or subsystem I/O generation to take effect.

**MSGSKEL** This SYSMOD contains message changes that must be compiled for translated versions of the message changes to become operational on extended TSO consoles.

If you want to use translated versions of the messages, you must run the message compiler once for the library containing the English message outlines, and once for each additional language you want to be available on your system. For details, see the *MVS/ESA Planning: Operations* manual.

If you want to use **only** the English version of the messages, you do not need to run the message compiler. You should bypass this reason ID.

**MVSCP** The SYSMOD requires the MVS configuration program to be run for the change to take effect.

**BYPASS(HOLDUSER)**

indicates that exception SYSMODs associated with the specified user reason IDs should not be held. The list of reason IDs is optional.

If you include a list of reason IDs, only the ones you specify are bypassed. If you do not include a list, all user reason IDs are bypassed.

### **BYPASS(ID)**

indicates that SMP/E should ignore any errors it detects when checking the SYSMOD's RMID and UMIDs. Generally, you should specify **BYPASS(ID)** on all APPLY CHECK commands to check for possible regressions.

### **BYPASS(IFREQ)**

indicates that SMP/E should ignore any conditional requisites that are missing. Generally, you should specify **BYPASS(IFREQ)** on all APPLY CHECK commands to check for possible regressions.

### **BYPASS(PRE)**

indicates that SMP/E should ignore any missing prerequisites. Generally, you should specify **BYPASS(PRE)** on all APPLY CHECK commands to check for possible regressions.

### **BYPASS(REQ)**

indicates that SMP/E should ignore any requisites that are missing. Generally, you should specify **BYPASS(REQ)** on all APPLY CHECK commands, to check for possible regressions.

### **CHECK**

indicates that SMP/E should not actually update any libraries. Instead, it should just do the following:

- Test for errors other than those that occur when the libraries are actually updated.
- Report on which libraries are affected.
- Report on any SYSMOD that would be regressed.

### **COMPRESS**

indicates which target libraries should be compressed. SMP/E does **not** compress any libraries that are actually paths in a hierarchical file system.

- If you specify **ALL**, any libraries in which elements will be installed by this APPLY command are compressed.
- If you specify particular ddnames, those libraries are compressed regardless of whether they will be updated.

#### **Notes:**

1. **COMPRESS** can also be specified as **C**.
2. If you specify both **COMPRESS** and **CHECK**, **COMPRESS** is ignored. This is because SMP/E does not update any data sets for **CHECK**.

### **EXCLUDE**

specifies one or more SYSMODs that should not be applied.

#### **Notes:**

1. **EXCLUDE** can also be specified as **E**.
2. SMP/E does not include a SYSMOD that would be included by the **GROUP** or **GROUPEXTEND** operand if that SYSMOD is specified on the **EXCLUDE** operand.

**EXSRCID**

indicates that SYSMODs associated with the specified source IDs should **not** be applied.

**Notes:**

1. There are two ways to specify source IDs:
  - Explicitly, by specifying a particular source ID (for example, **PUT9303**). In this case, only that particular source ID is used.
  - Implicitly, by specifying either **\*** or **c\*** (for example, **PUT\***), where *c* is a 1- to 7-character string. In the second case, all source IDs beginning with the specified character string are used.
2. A given source ID can be explicitly specified **only once** on the EXSRCID operand.
3. The same source ID **cannot** be explicitly specified on both the EXSRCID and source ID operands.
4. If a source ID is implicitly or explicitly specified on the EXSRCID operand and on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.
5. If a given SYSMOD has multiple source IDs, at least one of which is specified either implicitly or explicitly on the SOURCEID operand and another is specified on the EXSRCID operand, the SYSMOD is excluded from processing.  
  
For example, assume PTF UZ12345 has been assigned source IDs SMCREC and PUT9303. If you specify **SOURCEID(SMC\*)** and **EXSRCID(PUT9303)**, the SYSMOD is excluded from processing.
6. If a SYSMOD would be included by the GROUP or GROUPEXTEND operand, but is excluded by the EXSRCID operand, SMP/E does not include it.
7. If you do not specify any SYSMOD types, SMP/E processes only PTFs. To process other types of SYSMODs, you must specify the desired SYSMOD types.

**FORFMID**

indicates that only SYSMODs for the specified FMIDs or FMIDSETs should be applied.

**Notes:**

1. Functions containing a ++VER DELETE statement are not automatically included by the FORFMID operand. You must specify them on the SELECT operand.
2. If you do not specify any SYSMOD types, SMP/E processes only PTFs. To process other types of SYSMODs, you must specify the desired SYSMOD types.

## FUNCTIONS

indicates that all eligible functions should be applied.

### Notes:

1. **FUNCTIONS** can also be specified as **FUNCTION**.
2. If you specify **FUNCTIONS** along with **SELECT**, all eligible functions are included in addition to the SYSMODs specified on **SELECT**.
3. If you specify **FUNCTIONS** along with **SOURCEID**, all functions associated with the specified source IDs are included.
4. Functions containing a ++VER DELETE statement are not automatically included by the **FUNCTIONS** operand. You must specify them on the **SELECT** operand.

## GROUP

indicates that if any SYSMODs specifically defined as requisites for eligible SYSMODs have not yet been applied, SMP/E should automatically include them.

### Notes:

1. **GROUP** can also be specified as **G**.
2. **GROUP** is mutually exclusive with **GROUPEXTEND**.
3. **GROUP** may include SYSMODs at a service level higher than that specified by the **SOURCEID** operand.
4. If you specify **GROUP** with no other SYSMOD selection operands (such as a SYSMOD type, **SOURCEID**, **FORFMID**, or **SELECT**), **GROUP** is ignored.
5. Processing done for SYSMODs specified on the **SELECT** operand is not necessarily done for SYSMODs included by the **GROUP** operand. For example, if **REDO** is specified, only SYSMODs specified on the **SELECT** operand can be reapplied; SYSMODs included by the **GROUP** operand are not.
6. Functions containing a ++VER DELETE statement are not automatically included by the **GROUP** operand. You must specify them on the **SELECT** operand.
7. If a SYSMOD would be included by the **GROUP** operand, but is excluded by the **EXCLUDE** or **EXSRCID** operand, SMP/E does not include it.

## GROUPEXTEND

indicates that if a SYSMOD specifically defined as a requisite for an eligible SYSMOD has not been applied and cannot be processed for one of the reasons shown in Table 2 on page 61, SMP/E should automatically include a superseding SYSMOD. As the table shows, what **GROUPEXTEND** includes depends on why the requisite cannot be processed.



<i>Table 2. What GROUPEXTEND Includes (APPLY Processing)</i>	
<b>For a Requisite That Is:</b>	<b>GROUPEXTEND Includes:</b>
<ul style="list-style-type: none"> <li>• Held for an error reason ID</li> </ul>	<ul style="list-style-type: none"> <li>• A SYSMOD that supersedes the requisite OR</li> <li>• A SYSMOD that matches or supersedes the error reason ID</li> </ul>
<p><b>One</b> of the following:</p> <ul style="list-style-type: none"> <li>• Held for a system reason ID</li> <li>• Held for a user reason ID</li> <li>• Applied in error</li> <li>• Not available</li> </ul>	<ul style="list-style-type: none"> <li>• A SYSMOD that supersedes the requisite</li> </ul>

You can specify **NOAPARS** or **NOUSERMODS** (or both **NOAPARS** and **NOUSERMODS**) to limit the types of SYSMODS that are included by GROUPEXTEND to resolve error reason IDs. The default is to include all eligible SYSMODS, regardless of SYSMOD type.

**NOAPARS**

indicates that SMP/E should exclude APARs that resolve error reason IDs.

**NOUSERMODS**

indicates that SMP/E should exclude USERMODs that resolve error reason IDs.

**Notes:**

1. **GROUPEXTEND** can also be specified as **GEXT**.
2. **GROUPEXTEND** is mutually exclusive with **GROUP**.
3. If you specify both **BYPASS** and **GROUPEXTEND**, SMP/E does not include superseding SYSMODs needed to take the place of requisites that have been bypassed.  
  
During CHECK processing, if you want to see whether any superseding SYSMODs are available for requisites that have been bypassed, specify **GROUPEXTEND** without **BYPASS**.
4. **GROUPEXTEND** may include SYSMODs at a service level higher than what is specified by the **SELECT** or **SOURCEID** operands.
5. Functions and excluded SYSMODs are not automatically included by **GROUPEXTEND**.
6. Processing done for SYSMODs specified on the **SELECT** operand is not necessarily done for SYSMODs included by the **GROUPEXTEND** operand. For example, if **REDO** is specified, only SYSMODs specified on the **SELECT** operand can be reapplied; SYSMODs included by the **GROUPEXTEND** operand cannot.
7. If a SYSMOD would be included by the **GROUPEXTEND** operand, but is excluded by the **EXCLUDE** or **EXSRCID** operand, SMP/E does not include it.

- When **GROUPEXTEND** is specified, SMP/E examines more SYSMODs than it does if **GROUP** were specified. Because of this additional processing, the **APPLY** command runs longer than if **GROUP** was specified, and a larger region size may be needed.

On the other hand, **GROUPEXTEND** reduces the amount of time you would otherwise spend searching for missing requisites.

### **JCLINREPORT**

indicates that SMP/E is to write the JCLIN reports after processing inline JCLIN. This is the default.

**Note:** **JCLINREPORT** can also be specified as **JCLR**.

### **NOJCLIN**

indicates that SMP/E should not process inline JCLIN for the specified SYSMODs. For example, if you are reapplying SYSMODs, you may not want to process inline JCLIN that would change target zone entries that should not be changed.

If you include a list of SYSMOD IDs, SMP/E skips JCLIN processing only for the specified SYSMODs. If you do not include a list of SYSMOD IDs, SMP/E skips JCLIN processing for all SYSMODs.

### **NOJCLINREPORT**

indicates that SMP/E should not write any JCLIN reports after processing inline JCLIN.

**Note:** **NOJCLINREPORT** can also be specified as **NOJCLR**.

### **NUCID**

specifies the digit at the end of the IEANUC0n module under which the current nucleus is to be saved during **APPLY** processing. This value overrides the **NUCID** value specified in the **OPTIONS** entry that is in effect for this **APPLY** command.

### **PTFS**

indicates that all eligible PTFs should be applied.

#### **Notes:**

- PTFS** can also be specified as **PTF**.
- PTFS** is the default SYSMOD type for mass-mode processing. If you do not specify any other SYSMOD types, only PTFs are processed, even if **PTFS** was not specified.
- If you specify **PTFS** along with **SELECT**, all eligible PTFs are included in addition to the SYSMODs specified on **SELECT**.
- If you specify **PTFS** along with **SOURCEID**, all PTFs associated with the specified source IDs are included.

### **RC**

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the **APPLY** command.

Before SMP/E processes the **APPLY** command, it checks whether the return codes for the specified commands are less than or equal to the values speci-

fied on the RC operand. If so, SMP/E can process the APPLY command. Otherwise, the APPLY command fails. For more information about the RC operand, see Appendix D.

**Notes:**

1. The RC operand **must be the last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the APPLY command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

**REDO**

indicates that if any SYSMOD specified on SELECT has already been successfully applied, it should be reapplied.

**Notes:**

1. If you specify **REDO**, you must also specify **SELECT**.
2. If **GROUP** or **GROUPEXTEND** is also specified, REDO does not reapply SYSMODs included by the GROUP or GROUPEXTEND operand. It processes only SYSMODs specified on the SELECT operand.
3. If you use REDO to reapply a SYSMOD with inline JCLIN, you may not be able to restore that SYSMOD. This is the case if the target zone entries were updated the first time the SYSMOD was applied. When the SYSMOD is reapplied by use of the REDO operand, the target zone entries are first copied to the SMPSCDS as BACKUP entries, and then updated again for the SYSMOD. As a result, the BACKUP entries and the target zone entries are at the same level, and SMP/E has no record of the target zone entries before the SYSMOD was installed.

**RETRY**

indicates whether SMP/E should try to recover from out-of-space errors for utilities it calls.

**YES**

indicates that SMP/E should try to recover and retry the utility if a RETRYDDN list is available in the OPTIONS entry that is in effect. RETRY(YES) is the default.

If retry processing does not reclaim sufficient space and input to the utility was batched (copy or link-edit utility only), SMP/E debatches the input and retries the utility for each member separately. If this final attempt fails, the resulting x37 abend is treated as an unacceptable utility return code. In this case, processing continues for SYSMODs containing eligible updates to other libraries, but processing fails for SYSMODs containing unprocessed elements for the out-of-space library (and it fails for any SYSMODs that are dependent on the failed SYSMODs). For guidance on setting up the desired retry processing, see the *SMP/E R8.1 User's Guide*. For more information about OPTIONS entries, see "OPTIONS Entry (Global Zone)" on page 741.

If there is no RETRYDDN list, SMP/E does not try to recover from out-of-space errors, even if you specify **YES**.

### NO

indicates that SMP/E should not try to recover from the error.

### REUSE

indicates that if a module was successfully assembled during previous SMP/E processing, it should not be reassembled. Instead, the existing object module from SMPWRK3 should be reused.

### SELECT

specifies one or more SYSMODs that should be applied.

#### Notes:

1. **SELECT** can also be specified as **S**.
2. To reapply a SYSMOD, it is not enough to specify that SYSMOD on the **SELECT** operand. You must also specify **REDO**.
3. To process functions containing a ++VER DELETE statement, you must specify them on the **SELECT** operand.

### SOURCEID

indicates that SYSMODs associated with the specified source IDs should be applied.

#### Notes:

1. There are two ways to specify source IDs:
  - Explicitly, by specifying a particular source ID (for example, **PUT9303**). In this case, only that particular source ID is used.
  - Implicitly, by specifying either **\*** or **c\*** (for example, **PUT\***), where **c** is a 1- to 7-character string. In the second case, all source IDs that begin with the specified character string are used.
2. A given source ID can be explicitly specified **only once** on the **SOURCEID** operand.
3. The same source ID **cannot** be explicitly specified on both the **EXSRCID** and the **SOURCEID** operands.
4. If a source ID is implicitly or explicitly specified both on the **SOURCEID** operand and on the **EXSRCID** operand, all SYSMODs with that source ID are excluded from processing.
5. If a given SYSMOD has multiple source IDs, at least one of which is specified either implicitly or explicitly on the **SOURCEID** operand and another on the **EXSRCID** operand, the SYSMOD will be excluded from processing.  
  
For example, assume that PTF UZ12345 has been assigned source IDs **SMCREC** and **PUT9303**. If you specify **SOURCEID(SMC\*)** and **EXSRCID(PUT9303)**, the SYSMOD is excluded from processing.
6. Functions containing a ++VER DELETE statement are not automatically included by the **SOURCEID** operand. You must specify them on the **SELECT** operand.
7. If you do not specify any SYSMOD types, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.

**USERMODS**

indicates that all eligible USERMODs should be applied.

**Notes:**

1. **USERMODS** can also be specified as **USERMOD**.
2. If **USERMODS** is specified along with **SELECT**, all eligible USERMODs are included in addition to the SYSMODs specified on **SELECT**.
3. If **USERMODS** is specified along with **SOURCEID**, all USERMODs associated with the specified source IDs are included.

**Syntax Notes**

Figure 3 on page 66 shows how SMP/E chooses which SYSMODs to process, on the basis of the operands specified on the **APPLY** command.

- If you specify any of the operands in the top part of the chart, or if you do not specify the **SELECT** operand, SMP/E does *mass-mode* processing.
- If you specify the **SELECT** operand, SMP/E does *select-mode* processing.
- If you specify **SELECT** plus operands from the top part of the chart, SMP/E does both *select-mode* and *mass-mode* processing.

For more information about select-mode and mass-mode processing, see “Candidate Selection” on page 78.

Remember the following when coding the **APPLY** command:

1. SMP/E applies SYSMODs specified on **SELECT** regardless of other **APPLY** operands (such as a SYSMOD type, **SOURCEID**, **EXSRCID**, or **FORFMID**). Therefore, if you want to apply a specific SYSMOD, you only need to specify the SYSMOD ID on **SELECT**. For example, to apply a specific APAR, you do not also have to include the APAR operand.

**Note:** If you do specify a SYSMOD type along with **SELECT**, SMP/E applies all SYSMODs of the specified type plus the selected SYSMOD.

2. If you specify more than one SYSMOD type, a SYSMOD needs to match only one of the specified types.
3. If the **SOURCEID**, **FORFMID**, and SYSMOD type operands are specified together, only those SYSMODs meeting all the conditions are applied. (For a SYSMOD type, the SYSMOD must meet just one of the type conditions.)

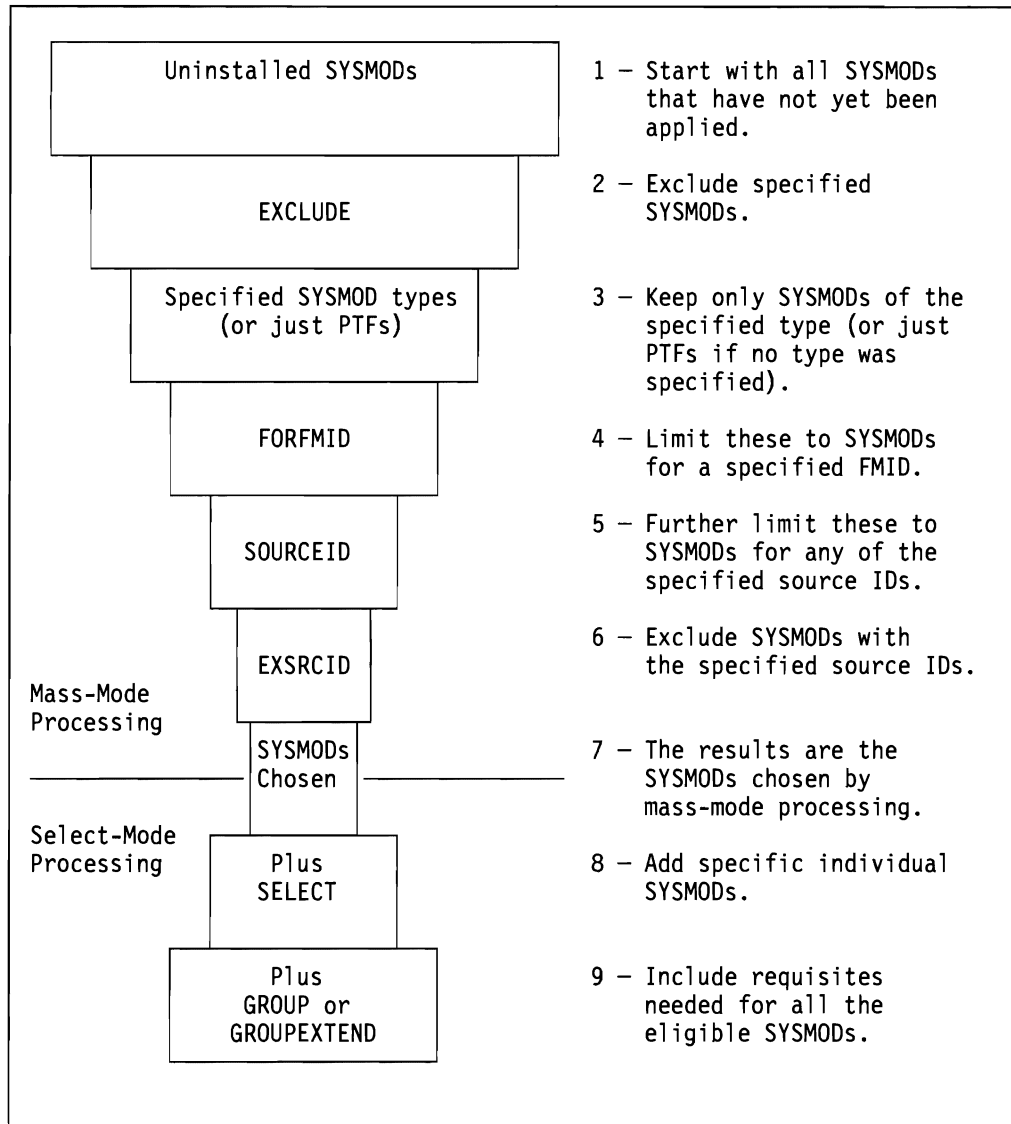


Figure 3. Combining SYSMOD Selection Operands on the APPLY Command

---

## Data Sets Used

The following data sets may be needed to run the APPLY command. They can be defined by DD statements or, ordinarily, by DDDEF entries. See Chapter 34 for more information about these data sets.

PARMLIB	SMPPTS	SMPWRK3	SYSUT4
SMP_CNTL	SMPRPTS	SMPWRK4	Distribution library
SMP_CSI	SMPSCDS	SMPWRK6	Link library
SMPLOG	SMP_SNAP	SYSLIB	Target library
SMPLOGA	SMPSTS	SYS_PRINT	Text library
SMPLTS	SMPTLIB	SYSUT1	<i>zone</i>
SMPMTS	SMPWRK1	SYSUT2	
SMPOUT	SMPWRK2	SYSUT3	

### Notes:

1. The SMPLTS data set is required only if a load module having a CALLIBS subentry list is being created, updated, deleted, or renamed.
2. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are allocated dynamically by use of the ZONEINDEX information in the GLOBALZONE entry.

---

## Usage Notes

This section discusses the following:

- “Adding New Elements Other Than Modules to the Target Libraries”
- “Adding New Modules to the Target Libraries” on page 68
- “Checking the DISTLIB Operand” on page 68
- “DISTSRC, ASSEM, and DISTMOD Operands” on page 69
- “Use of the SMPMTS and SMPSTS as Target Libraries” on page 70
- “Use of the SMPLTS Library” on page 70
- “Alias Processing” on page 70
- “APPLY CHECK Facility” on page 71
- “SYSMOD Termination” on page 71
- “APPLY Termination” on page 72
- “Automatic Reapplication of SYSMODs” on page 73

## Adding New Elements Other Than Modules to the Target Libraries

The target library for macros, source, data elements, and HFS elements is found as a SYSLIB subentry for an element entry in the target zone. If no SYSLIB subentry is present and no SYSLIB is specified on the element's MCS, a check is made to determine whether the element's distribution library has been totally copied to a target library. If it has (as shown by the presence of a target zone DLIB entry for the distribution library), the target library to which the DLIB was copied is determined to be the target library for the element.

**Note:** There should be only one SYSLIB subentry for the DLIB entry, and the SYSLIB subentry must specify the ddname of the target library for the elements.

### Adding New Modules to the Target Libraries

The SMP/E APPLY process can be used to install a SYSMOD that introduces a new module into the system libraries. To apply a module for the first time, SMP/E requires that the DISTLIB operand be specified on the ++MOD statement.

If the module is to be installed into an existing load module, this fact can be identified to SMP/E in two ways:

- If no additional link-edit control statements (other than the INCLUDE statement) are required to rebuild the load module, the LMOD operand on the ++MOD statement can be used to indicate that the new module is to be link-edited into the existing load module. If no LMOD entry exists in the target zone for the specified load module, an error results.
- If adding the new module requires the addition of a link-edit control card in order to rebuild the load module (for example, another ORDER card is required or a new ENTRY is established), inline JCLIN is required to redefine the load module structure.

If the module is part of a copied library, SMP/E already has all the information necessary to install the module fully; therefore, no special operands or processing is required. SMP/E uses the DLIB entry to determine that the distribution library has been totally copied, to build a new target zone MOD entry with a LMOD subentry equal to the module name, and to build a LMOD entry (with a name equal to the module name), using the SYSLIB value from the DLIB entry and the LEPARMS from the ++MOD LEPARMS.

### Checking the DISTLIB Operand

When an element is selected for application and a target zone entry for that element already exists, the value of the DISTLIB operand on the element is compared with the DISTLIB subentry in the target zone element entry. If they are not equal, SMP/E issues a message to inform you of an error condition and terminates the SYSMOD containing the element.

If service and function SYSMODs are being processed and contain the same element, and no element entry exists in the target zone, the service SYSMODs must specify the same DISTLIB as the function SYSMODs on the elements. If they do not, SMP/E issues an error message and the APPLY command is terminated.

If two service SYSMODs update or replace the same element, have different DISTLIB operand values, and are both eligible for processing, but no entry for the element exists in the target zone, then the first SYSMOD processed causes an element entry to be created in the target zone with the element entry containing the DISTLIB from its MCS statement. SMP/E terminates the second SYSMOD.

**Note:** The DISTLIB is not checked for function SYSMODs, because the SYSMOD defines the initial distribution libraries for its elements.



## DISTSRC, ASSEM, and DISTMOD Operands

Because SMP/E cannot determine from the data processed by JCLIN what sources are contained in a totally copied library, the DISTSRC, ASSEM, and DISTMOD operands are provided to pass this information to SMP/E when a macro is replaced or updated, and the macro change must cause the source to be reassembled.

- The DISTSRC operand value specifies the name of the distribution library containing the source.
- The ASSEM and PREFIX operand values specify a list of ASSEM entries in the source or target zone that should be assembled during APPLY processing.
- The DISTMOD operand value specifies the name of the distribution library containing the load modules.

These four operands are specified on ++MAC and ++MACUPD statements. The DISTMOD operand is also specified on ++SRC and ++SRCUPD statements.

The ASSEM operand values are placed in the associated SYSMOD entry on the target zone as ASSEM subentries. If any of the modules specified in the ASSEM operand values are found on the target zone as SRC or ASSEM entries, the DISTLIB and SYSLIB subentry values are used in lieu of the DISTSRC operand value.

If neither a SRC nor an ASSEM entry exists for a module in the ASSEM operand values, a SRC entry is created. The DISTSRC operand value is placed in the SRC entry as the DISTLIB subentry. If there is a DLIB entry in the target zone for the DISTSRC operand value, the SYSLIB subentries from the DLIB entry are placed in the SRC entry as SYSLIB subentries. If no DLIB entry exists, the SYSLIB subentry in the SRC entry is left null, and the SMPSTS is used in place of a target library.

If there is no MOD entry in the target zone for a module in the ASSEM operand list, one is created. The DISTMOD operand value is placed in the MOD entry as the DISTLIB subentry.

If no LMOD entry exists for a module, one is created, provided the target zone contains a DLIB entry for the DISTMOD operand value. The SYSLIB subentries from the DLIB entry are placed in the LMOD entry as SYSLIB subentries, and the LMOD subentry is placed in the MOD entry. If no DLIB entry exists, no LMOD subentry exists in the MOD entry, and, therefore, no executable load module can be updated in the target system for that module.

After the macro has been updated or replaced, all the modules specified in the ASSEM and PREFIX operand lists are assembled. If no member is found in the necessary library (the source target system library, the SMPSTS, or the distribution library) for a source specified in the ASSEM operand list, SMP/E issues a warning message and goes on processing the SYSMOD without assembling or link-editing the module. If an assembly completes with a return code greater than the one that you specified in the RC subentry of the ASM UTILITY entry (or the SMP/E default of 4, if the RC subentry is null), the processing of the SYSMOD is terminated. If the resulting object text from a successful assembly can be link-edited into a load module, the link-edit is performed.

### Use of the SMPMTS and SMPSTS as Target Libraries

If no operating system library can be determined, macros are stored in the SMPMTS library and the source in the SMPSTS library.

Macro and source elements stored in the SMPMTS or SMPSTS remain there until they are replaced by elements from subsequent APPLY processing or until the SYSMODs that modified them are accepted. In this way, the SMPMTS serves as a macro library for assemblies during APPLY processing and must be in the concatenation of the SYSLIB DD statement for APPLY. Likewise, the SMPSTS serves as a source library for assemblies during APPLY processing, but is **not** required in the SYSLIB concatenation.

### Use of the SMPLTS Library

The SMPLTS data set is a target library used to maintain the “base” version of a load module specifying a SYSLIB allocation in order to implicitly include modules. Such a load module is built in two stages:

1. The “base” version of the load module is built in the SMPLTS data set. This version contains only modules that are explicitly defined in the link-edit JCL as being included in the load module.
2. The executable version of the load module is built in the target libraries using the load module's SYSLIB allocation (the CALLLIBS subentry list in its LMOD entry) and the “base” version of the load module from the SMPLTS data set. This version contains modules that are implicitly defined through the SYSLIB allocation as being included in the load module. If the SMPLTS does not contain the “base” version of the load module, the executable version is not built in the target libraries.

A load module stored in the SMPLTS remains there until it is replaced by a new version from subsequent APPLY processing.

For more information, see “Building Load Modules with a SYSLIB Allocation” on page 99.

### Alias Processing

When an element with aliases is processed, both the element and its aliases are updated. SMP/E does not check the aliases against elements maintained in the target zone. The user must make sure an element's alias does not match the name of an element maintained by SMP/E in the target zone.

Aliases for an element are determined as follows:

- Replacement elements (MACs, MODs, and data elements):
  - If a list of aliases is specified on the SMP/E MCS, these aliases are used. Aliases (on the target system) existing before this new list of aliases was presented to SMP/E are not replaced. Further, this new list of aliases replaces any alias subentries in the target zone element entry.
  - If no list of aliases is specified on the SMP/E MCS, the aliases found as alias subentries in the target zone element entry are used.

- Update elements (ZAPs and MACUPDs):
  - If a list of aliases is specified on the SMP/E MCS, these aliases are used. Any alias subentries in the target zone element entry are ignored for update processing of the element. Macro aliases (on the target system) existing before this list of aliases was presented to SMP/E are not updated. Alias subentries in the target zone element entry are not updated or replaced by the aliases in this list.
  - If no list of aliases is specified on the SMP/E MCS, the aliases found as alias subentries in the target zone element entry are used.

## APPLY CHECK Facility

The purpose of the CHECK option is to perform a dry run to inform you of possible errors, and to provide reports of SYSMOD status, libraries that will be updated, regression conditions, and SYSMODs that will be deleted. Target system libraries are not permanently updated.

During check processing, the list of target zone entries is maintained in storage; data is written to the target zone as a temporary storage medium. Check processing deletes any data written to the target zone. Consequently, no permanent updates are made to the target zone.

## SYSMOD Termination

Termination of a SYSMOD causes a return code of 8. Termination of a ++FUNCTION causes a return code of 12. Termination occurs in response to any of the following conditions:

- Missing requisites:
  - The requisite SYSMOD is not available on the PTS/CSI data sets. (It has not been received.)
  - The requisite SYSMOD has been excluded.
  - The requisite SYSMOD was terminated (possibly because other requisites are missing).
  - The requisite SYSMOD did not meet the applicability criteria.
  - The requisite SYSMOD was not included in the SELECT list, and neither **GROUP** nor **GROUPEXTEND** was specified.
  - **GROUP** was specified to include the requisite, but the requisite SYSMOD is being held or is not available on the PTS or CSI data sets. (It has not been received.)
  - **GROUPEXTEND** was specified to supersede the failing requisite, but a superseding SYSMOD was not available for processing.
- Inline JCLIN processing failure. The entries affected are restored to the state that existed before JCLIN processing.
- MODID error conditions.
- Attempting to change the ownership of an element that is being updated rather than replaced.
- DISTLIB operand checking failure.

- DD statement missing for a target system library.
- Utility return codes. Return codes from the utilities called to update, assemble, copy, and link-edit elements to the target system are examined to determine whether the operation has succeeded or failed. If these return codes exceed a predefined value, the SYSMODs whose elements are involved in the operation are terminated. For details on handling x37 abends, see the description of the RETRY operand under “Operands” on page 55.
- Related SYSMOD failure. When SMP/E excludes an element from a SYSMOD because another SYSMOD being processed supplies a higher level of the element, SMP/E does not consider the first SYSMOD successfully processed until the SYSMOD supplying the highest (selected) level element completes successfully. If the SYSMOD supplying the highest level element fails, all SYSMODs from which elements have been excluded are terminated because of a “related SYSMOD failure.”

### Avoiding SYSMOD Termination

**BYPASS:** Certain error conditions causing the termination of a SYSMOD can be avoided by specifying the BYPASS operand on the APPLY command. In BYPASS mode, some error conditions are treated as warning conditions. The following operand values can be specified with the BYPASS operand to avoid termination:

<b>ID</b>	Specifies that SYSMODs should be processed even though their MODID verification checks have failed.
<b>PRE</b>	Specifies that SYSMODs should be processed even though their PRE requisite conditions are not met.
<b>REQ</b>	Specifies that SYSMODs should be processed even though their REQ requisite conditions are not met.
<b>IFREQ</b>	Specifies that SYSMODs should be processed even though their conditional requisite conditions (IFREQs) are not met.

**Utility Return Code Thresholds:** The value SMP/E uses to determine the success or failure of a called utility program is kept in the UTILITY entries and can be changed by UCLIN.

### APPLY Termination

APPLY processing termination causes a return code of 12. For each of the following conditions, SMP/E issues an error message. APPLY reports are not produced when a function SYSMOD is terminated before selection processing completes. Termination can be caused by any of the following conditions:

- Termination of processing of any function SYSMOD.
- Two function SYSMODs are specified in the SELECT list, and one specifies the other in the DELETE operand of its ++VER statement.
- Two function SYSMODs are specified in the SELECT list or are selected in mass mode, and one specifies the other in the NPRES operand of its ++VER statement.
- A function SYSMOD specifying a previously applied SYSMOD in the NPRES operand of its ++VER statement is specified in the SELECT list.

- A function SYSMOD deleted by a previously applied SYSMOD (that is, a SYSMOD entry in the target zone indicates that the SYSMOD has been deleted) is specified in the SELECT list.
- A function SYSMOD superseded by a previously applied SYSMOD (that is, a SYSMOD entry in the target zone indicates that the SYSMOD is superseded) is specified in the SELECT list. A service SYSMOD in the same situation is not processed, but the APPLY command is not terminated.

## Automatic Reapplication of SYSMODs

An applied SYSMOD can be selected for reapplication when a function SYSMOD is applied for the first time. This can occur if the modification is applicable to more than one function. For example, consider the following SYSMOD:

```
++PTF(UZ00001).  
++VER(Z038) FMID(GVT3100).  
++IF      FMID(GVT3101) THEN REQ(UZ00001).  
++VER(Z038) FMID(GVT3101).  
++MOD(IFTABCD) DISTLIB(AOS99).
```

If this PTF was first applied when only function GVT3100 was installed, the first ++VER statement would have been used, and the conditional requisite data supplied on the ++IF would have been saved. If GVT3101 is installed later, the saved ++IF data requires this same PTF to be installed.

**Note:** Because SMP/E does not process a SYSMOD with more than one VER that appears to be valid, GVT3101 must DELETE GVT3100 in order for this construction to work properly.

---

## Output

The following reports can be produced during APPLY processing:

- Causer SYSMOD Summary report
- Cross-Zone Summary report
- Deleted SYSMOD report
- File Allocation report
- Element Summary report
- JCLIN Cross-Reference report
- JCLIN Summary report
- MOVE/RENAME/DELETE report
- SYSMOD Regression report
- SYSMOD Status report

See Chapter 31 for descriptions of these reports.

---

## Examples

The following examples are provided to help you use the APPLY command:

- “Example 1: Applying All SYSMODs from a Given Source” on page 74
- “Example 2: Applying All SYSMODs for Selected Functions” on page 74
- “Example 3: Applying APARs and USERMODs” on page 74
- “Example 4: Applying with the GROUP Operand” on page 75
- “Example 5: Applying with GROUPEXTEND” on page 75

- “Example 6: Applying with the CHECK Operand” on page 76
- “Example 7: Combining APPLY Operands” on page 76
- “Example 8: Installing Service for All ESO Service Levels” on page 77
- “Example 9: Excluding SYSMODs with Certain Source IDs” on page 77

### Example 1: Applying All SYSMODs from a Given Source

If the SOURCEID operand was used during RECEIVE processing to group all those SYSMODs processed, you can choose to install only that set of SYSMODs. This can be done with the SOURCEID operand of the APPLY command. Suppose you received an ESO containing service levels PUT9301 and PUT9302. The ESO contained ++ASSIGN statements that assigned each PTF a SOURCEID value corresponding to the service level that it is part of. Now you want to install all the applicable PTFs from those tapes into the target libraries described by zone MVSTST1. You can do this with the following commands:

```

SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
APPLY    SOURCEID(PUT9301, /* Process these service */
          PUT9302) /* levels */
          GROUP           /* and any requisites. */.
    
```

### Example 2: Applying All SYSMODs for Selected Functions

At times, you may want to install changes only for a single function or for a certain group of functions. This can be done with the FORFMID operand of the APPLY command. Assume you want to install service for function JXX1234 and for all the telecommunication-related functions on your system. You first need to define an FMIDSET for the telecommunication functions. You can do this with the following commands:

```

SET      BDY(GLOBAL)      /* Process global zone. */.
UCLIN    /* UCLIN to set up
          FMIDSET. */.
ADD      FMIDSET(TC)      /* Define TC FMIDSET. */
          FMID(JXX0001    /* Include these FMIDs. */
          JXX0002) /* */.
ENDUCL   /* End UCL set up. */.
    
```

You can now use the following commands to install PTFs for function JXX1234 and for the functions in FMIDSET TP:

```

SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
APPLY    FORFMID(JXX1234, /* Apply for selected FMIDs. */
          TC) /* */.
    
```

### Example 3: Applying APARs and USERMODs

You may want to install just corrective fixes (APAR fixes) or user modifications (USERMODs) into the target libraries. This can be done with the APARS and USERMODS operands of the APPLY command. Assume you want to install all APAR fixes and USERMODs for function HXY2102. You can do this with the following commands:

```

SET      BDY(MVSTGT1)      /* Process MVSTGT1 tgt zone. */.
APPLY    FORFMID(JXY2101) /* Apply for this FMID */
          APARS           /* all APARs */
          USERMODS       /* and all USERMODs. */.
    
```

The APARS and USERMODS operands indicate that SMP/E is to pick only APAR fixes and USERMODS.

**Note:** If you want to install specific APAR fixes or USERMODS, use the SELECT operand. You do not need to specify APARS or USERMODS, which install **all** SYSMODs of the specified type.

### Example 4: Applying with the GROUP Operand

At times, you may know that a particular SYSMOD is required on your system, but you may not know all its requisite SYSMODs. By using the GROUP operand of APPLY, you can have SMP/E determine all the requisites and automatically install them. This method is often used during the installation of new functions. Suppose you want to install a new function, HYY2102, plus all its service, plus any requisite SYSMODs. You can do this with the following commands:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
APPLY    FORFMID(HYY2102) /* For one function.          */.
          FUNCTIONS PTFs   /* Function and PTFs         */.
          GROUP            /* plus requisites.          */.
```

The FORFMID operand indicates that only SYSMODs applicable to this function should be installed. The FUNCTIONS operand indicates that HYY2102 can be installed. The PTFs operand indicates that only PTFs for HYY2102 should be installed (no APAR fixes or USERMODS are included). The GROUP operand indicates that **all** requisite SYSMODs should also be applied. These requisites can be applicable to other functions, but cannot be APAR fixes or USERMODS.

### Example 5: Applying with GROUPEXTEND

Assume you want to use have SMP/E automatically include the requisites for some SYSMODs you plan to install. However, you are not sure whether all the requisites are available. (They might not be received, or they may be held because they are in error.) In those cases, you want SMP/E to check whether a superseding SYSMOD is available for the unsatisfied requisites. To have SMP/E do this additional checking, you can use the GROUPEXTEND operand:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
APPLY    FORFMID(HYY2102) /* For one function.          */.
          FUNCTIONS PTFs   /* Functions and PTFs plus   */.
          GROUPEXTEND      /* requisites or supersedes. */.
```

SMP/E applies HYY2102 and any functions or PTFs applicable to HYY2102. Because of the GROUPEXTEND operand, SMP/E also applies all requisites for those SYSMODs, even those not applicable to HYY2102. If SMP/E cannot find a requisite, it looks for a SYSMOD superseding the requisite and uses it to satisfy the requirement. Likewise, if a requisite is held for an error reason ID, SMP/E looks for a SYSMOD superseding the requisite, or that either satisfies or supersedes that error reason ID, and uses it to satisfy the requirement.

## Example 6: Applying with the CHECK Operand

In Example 4, SMP/E was directed to automatically include SYSMODs needed for the selected function and service. At times, you may want to review which SYSMODs is included before you actually install them. This can be done by using the CHECK operand of APPLY, as in the following commands:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
APPLY    FORFMID(HYY2102) /* For one FMID. */.
          FUNCTIONS PTFS  /* Functions and PTFs */.
          GROUP           /* plus requisites */.
          CHECK           /* in check mode. */.
```

After running this command, check the SYSMOD Status report to see which SYSMODs would have been installed if you had not specified CHECK. If the results of this trial run are acceptable, run the commands again, without the CHECK operand, to actually install the SYSMODs.

## Example 7: Combining APPLY Operands

If you want to further divide the work that is to be done, you can specify combinations of the APPLY operands. The following is an example using all the SYSMOD selection operands of APPLY:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
APPLY    SOURCEID(PUT9301 /* For these service levels. */.
          PUT9302) /* */.
          FORFMID(HYY2102 /* For selected functions */.
          TP) /* */.
          FUNCTIONS PTFS /* install all type SYSMODs */.
          APARS USERMODS /* */.
          SELECT (UZ00001 /* plus these three for */.
          UZ00002 /* other functions, */.
          UZ00003) /* */.
          EXCLUDE(UZ00010 /* but not these three, */.
          UZ00011 /* */.
          UZ00012) /* */.
          GROUP           /* plus all requisites. */.
```

This APPLY command causes SMP/E to apply all the SYSMODs that came from either service level PUT9301 or PUT9302 (from the SOURCEID operand) and that are applicable either to function SYSMOD HYY2102 or to one of the function SYSMODs identified in the FMIDSET entry "TP" (specified on the FORFMID operand), plus any other SYSMODs required to install those SYSMODs (specified on the GROUP operand). All SYSMOD types are eligible for selection. (This includes the FUNCTIONS, PTFS, APARS, and USERMODS operands.) In addition, the selected SYSMODs (UZ00001, UZ00002, and UZ00003) are applied even though they came from a source other than the two specified service levels and even though they may belong to function SYSMODs other than those specified (from the SELECT operand). SMP/E does not install SYSMODs UZ00010, UZ00011, or UZ00012, even though they may be applicable to the functions specified and have one of the two SOURCEID values or may be required to install other SYSMODs that are eligible (from the EXCLUDE operand).



## Example 8: Installing Service for All ESO Service Levels

Assume you want to install the preventive service received from all ESO tapes into zone MVSESA1 without having to specify all possible ESO service levels on the SOURCEID operand. You can use the following commands:

```

SET      BDY(MVSESA1)      /* Process MVSESA1 tgt zone. */.
APPLY    PTFS              /* Install all PTFS          */.
          SOURCEID(PUT*)   /* for all service levels   */.
          CHECK            /* in check mode.          */.

```

## Example 9: Excluding SYSMODs with Certain Source IDs

Assume you have received an ESO with PTFS up to service level PUT9303, and now you want to install service from all but the latest two service levels (PUT9302 and PUT9303) into zone MVSESA2. You can use the following commands:

```

SET      BDY(MVSESA2)      /* Process MVSESA2 tgt zone. */.
APPLY    PTFS              /* Install all PTFS          */.
          SOURCEID(PUT*)   /* for all service levels   */.
          EXSRCID(PUT9302 /* except for PUT9302     */.
                PUT9303) /* and PUT9303,           */.
          GROUPEXTEND      /* and any requisites     */.
          CHECK            /* in check mode.          */.

```

## Processing

APPLY processing is very similar to ACCEPT processing except that the target zone, rather than the distribution zone, controls processing, and the target libraries, rather than the distribution libraries, are updated.

## SYSMOD Selection

This section outlines the process by which SYSMODs and the elements from the SYSMODs are selected.

### Operands Related to SYSMOD Selection

The following APPLY command operands can be used to specify to SMP/E which SYSMODs are to be processed:

```

APARS
EXCLUDE
EXSRCID
FORFMID
FUNCTIONS
GROUP
GROUPEXTEND
PTFS
SELECT
SOURCEID
USERMODS

```

### Candidate Selection

The SYSMOD selection operands of the APPLY command can be specified separately or in combination to control the SYSMODs that SMP/E is to process. When you specify them separately, SMP/E selects only SYSMODs meeting the one criterion specified. When you specify them in combination, SMP/E does the following checking to build the complete candidate list:

1. SMP/E checks the global zone and the specified target zone to determine which SYSMODs present in the global zone and SMPPTS have not already been applied to the target zone. For each such SYSMOD, SMP/E checks to see if it meets the criteria of any additional selection operands.
  - a. If the EXCLUDE operand was specified, SMP/E makes sure the SYSMOD was not specified in the exclude list.
  - b. If one or more of the SYSMOD type operands (that is, FUNCTIONS, PTFS, APARS, or USERMODS) were specified, SMP/E checks to make sure the SYSMOD type was one of those specified. If no type operand was specified, the default is for SMP/E to process only PTF SYSMODs.
  - c. If the FORFMID operand was specified, SMP/E makes sure that either the FMID value on one of the ++VER statements within the SYSMOD or the SYSMOD ID itself matches either an FMID specified in FORFMID or one of the FMID values in a specified FMIDSET.
  - d. If the SOURCEID operand was specified, SMP/E makes sure one of the source IDs of the SYSMOD matches a source ID that was either explicitly or implicitly specified on the SOURCEID operand.
  - e. If the EXSRCID operand was specified, SMP/E makes sure none of the source IDs of the SYSMOD matches a source ID that was either explicitly or implicitly specified on the EXSRCID operand.

**Note:** If a given SYSMOD has multiple source IDs and you specify at least one of them, implicitly or explicitly, on the SOURCEID operand, and you specify another one implicitly or explicitly, on the EXSRCID operand, that SYSMOD is excluded from processing.

Similarly, if a given source ID is implicitly or explicitly specified on the EXSRCID operand and also on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.

Each SYSMOD satisfying all the above conditions is a candidate for the APPLY process. In other words, specifying the SYSMOD type operands, the FORFMID operand, or the SOURCEID operand in combination causes SMP/E to select those SYSMODs meeting all the specified conditions.

2. If you specify the SELECT operand, each SYSMOD specified in the select list is a candidate regardless of its SYSMOD type, FMID value, or SOURCEID value. That is, SELECT has an additive effect on the SYSMOD selection. This is called *select-mode* processing. If SELECT is not specified, this is called *mass-mode* processing.

**Note:** If SELECT is the only specified operand, SMP/E processes only the SYSMODs in the select list.

3. If the GROUP or GROUPEXTEND operand was specified, SMP/E checks each of the candidate SYSMODs to determine if it has any requisites that must also be applied. SMP/E automatically includes the following SYSMODs, as long as

they were not specified on the EXCLUDE operand or excluded by the EXSRCID operand:

- a. Any SYSMOD not already applied and specified as a prerequisite (that is, specified in the ++VER statement PRE operand) of one of the candidate SYSMODs
- b. Any SYSMOD not already applied and specified as a corequisite (that is, specified in the ++VER statement REQ operand) of one of the candidate SYSMODs
- c. Any SYSMOD not already applied and specified as a conditional requisite (that is, specified in the ++IF statement REQ operand) of one of the candidate SYSMODs
- d. Any SYSMOD not already applied and specified as a conditional requisite (CIFREQ subentry) in the target zone SYSMOD entry for one of the candidate SYSMODs

If one of these requisites is held or not available, and you specified **GROUPEXTEND**, SMP/E checks the global zone for any SYSMODs that have been received and that either supersede the requisite, or satisfy or supersede its HOLDERERROR reason ID. If so, the lowest-level SYSMOD found is used to satisfy the requisite.

- If you specified **NOAPARS** with **GROUPEXTEND**, SMP/E does not include APARs that resolve error reason IDs for the held requisites.
- If you specified **NOUSERMODS** with **GROUPEXTEND**, SMP/E does not include USERMODs that resolve error reason IDs for the held requisites.

Once a SYSMOD is added to the candidate list, it is eligible to be checked for additional requisites. The FORFMID, SOURCEID, and SYSMOD type operands have no effect on SYSMODs brought in because of the GROUP or GROUPEXTEND operand. The following example applies all those SYSMODs with a source ID of PUT9301 that are applicable to EBB1102, plus any additional SYSMODs that are required, even though their source ID is not PUT9301 or their FMID is not EBB1102:

```
SET      BDY(TGT1)          /* Set to target zone. */.
APPLY    SOURCEID(PUT9301) /* */.
          FORFMID(EBB1102) /* */.
          GROUP            /* */.
```

### Applicability Checking

Once the APPLY candidate list is completed, SMP/E does the following checking to make sure the selected SYSMODs are applicable to the system:

**General Applicability:** SMP/E makes sure that each SYSMOD meets certain requirements before it is applied to the system.

1. Each SYSMOD must contain **at least one** ++VER statement whose SREL value matches one of the SREL values for that target zone and whose FMID value (if present) names a function SYSMOD that has been (or is being) applied. Function SYSMODs having no ++VER statement FMID operand are applicable if the SREL matches.

Each SYSMOD must have **at most one** ++VER statement whose SREL matches the SREL in the target zone entry and whose FMID value exists in the target zone (or is being applied) and has not been superseded.

If a SYSMOD is found containing multiple applicable ++VER statements, SMP/E is unable to apply that SYSMOD, because SMP/E cannot determine which ++VER statement to use.

**Note:** SMP/E does not consider an FMID that has been superseded to be applied. Therefore, a SYSMOD can contain two ++VER statements that apply to two functions, one of which supersedes the other.

2. The SYSMOD must not have already been applied successfully to the target zone.
  - To reapply a SYSMOD, you must specify it in the SELECT operand and include the REDO operand on the APPLY command.
  - SYSMODs that have been partially applied during a previous invocation are considered eligible for processing without any special action. These SYSMODs are identified by the APPLY ERROR indicator in their target zone SYSMOD entry.
3. The SYSMOD must not have been partially restored during a previous SMP/E RESTORE attempt. These SYSMODs are identified by the RESTORE ERROR indicator in their target zone SYSMOD entry.
4. The SYSMOD must not be superseded by a previous SYSMOD. If a SYSMOD is found to be superseded by another SYSMOD being applied, no elements are selected from the superseded SYSMOD.
5. The SYSMOD must not have been explicitly deleted by a previous SYSMOD.

**Unconditional Requisites (PRE and REQ):** Unconditional requisites are SYSMODs that are required in all functional environments. All the unconditional requisites for each SYSMOD must be satisfied. Unconditional requisites are those specified in the SYSMOD's ++VER statement PRE and REQ operands. A requisite is considered satisfied if:

- The requisite SYSMOD is already applied.
- The requisite SYSMOD is superseded by a SYSMOD that is already applied.
- The requisite SYSMOD is being applied.
- The requisite SYSMOD is superseded by a SYSMOD being applied.

**Conditional Requisites (IFREQ):** Conditional requisites are SYSMODs required only for a particular functional environment. All conditional requisites of each SYSMOD must be resolved. Conditional requisites are specified on the ++IF statement immediately following the applicable ++VER statement. If the function specified in the FMID operand of the ++IF statement is applied or is being applied, each SYSMOD specified in the ++IF statement REQ operand must be satisfied. These requisites are satisfied in the same manner as unconditional requisites.

If the function specified in the FMID operand of the ++IF statement is not already installed, in order to make sure these requisites are satisfied when the function is installed, SMP/E saves the information from the ++IF statement as CIFREQ subentries in the target zone SYSMOD entry for that function. When the function is applied, SMP/E checks the CIFREQ subentries for requisites supplied by previously applied SYSMODs and makes sure these requisites are satisfied.

**Negative Requisites (NPRE):** If the NPRE operand is specified on a SYSMOD's ++VER statement, the SYSMOD ID specified must not already be installed, must not be installed concurrently, and must not be superseded by a SYSMOD being installed concurrently.

**Note:** The NPRE operand is valid only in function SYSMODs and is used to specify one or more mutually exclusive functions.

**Superseding SYSMODs (SUP):** SMP/E checks to make sure that the SYSMOD has not been superseded by another SYSMOD that is already installed or by another SYSMOD being applied concurrently. If the SYSMOD is superseded, it is not applied, and the superseding SYSMOD is used instead.

**Note:** If the superseding SYSMOD is not processed because it is held or excluded or because it has missing requisites, processing continues as though the SYSMOD did not exist. The SYSMOD that would have been superseded is installed instead.

**Exception SYSMODs (HOLD):** SMP/E makes sure that each SYSMOD has no unresolved exception data (that is, HOLD reason IDs) associated with it. Exception data is information specified on the ++HOLD statement. Each ++HOLD statement has a REASON operand specifying a character string identifying the reason the SYSMOD is being put into exception status. SMP/E supports the following types of exception data:

- HOLDERROR
- HOLDSYSTEM
- HOLDUSER

The ++HOLD statement can also contain a CLASS operand, which indicates an alternative way to resolve a reason ID.

Exception data is considered resolved when one or more of the following are true:

- The reason ID associated with the exception is found as a SYSMOD entry in the target zone.
- The reason ID associated with the exception is being applied concurrently or is being superseded by a SYSMOD being applied concurrently.
- The applicable BYPASS operand (HOLDERROR, HOLDSYSTEM, HOLDUSER, or HOLDCLASS) was specified.

Unless **all** the reason IDs associated with a given SYSMOD are resolved (and an appropriate HOLDCLASS is specified), SMP/E does not apply that SYSMOD. The SYSMOD is treated as though it had been specifically excluded. SMP/E also makes sure no SYSMOD is installed that is dependent on another SYSMOD in hold status. Messages are issued showing what reason IDs were not resolved, and the SYSMOD and reason IDs are displayed in the SYSMOD Summary report.

Each category of exception data is resolved differently:

- An **error reason ID** is actually the number of the APAR that caused the SYSMOD to be placed in exception status. As subsequent service is processed, the APAR will probably be superseded by a PTF. When this happens, the error reason ID is resolved and the first PTF is automatically processed.

Therefore, it is generally not necessary to use the BYPASS operand to process SYSMODs with error reason IDs.

During any mass application of SYSMODs, you should expect some SYSMODs not to be applied because unresolved APARs are associated with them. During the application of preventive service, such a SYSMOD should not be investigated further; it will be installed later when a subsequent SYSMOD is produced that supersedes the reason ID that is holding it.

During the application either of corrective service (that is, installing a PTF or an APAR because of a known problem in the system) or of a new function that specifically requires a SYSMOD, the reason IDs listed should be used as the first piece of data for research. Research may provide a fix for the problem, in which case the SYSMOD and the fix can be applied concurrently. However, if a fix is not available, you can either wait for one, or apply the SYSMOD using the appropriate BYPASS operand.

- A **system reason ID** is a 1- to 7-character string used to identify some action that must be taken before or after installation of a SYSMOD. System reason IDs are not SYSMOD IDs and are not specified in the supersede list of a SYSMOD. Therefore, SMP/E does not release them automatically.

In applying SYSMODs with system reason IDs, you should use the BYPASS(HOLDSYS(*reason-id*)) operand. If you were to remove the system reason ID by use of the ++RELEASE statement, you would be able to install the SYSMOD, but you would also lose the information about any special processing required if you were to try to apply that SYSMOD on another system.

**Note:** If the ++HOLD statement with the system reason ID was specified inline (within the held SYSMOD) instead of in the SMPHOLD data set, the **only** way to remove the system reason ID is to specify the BYPASS(HOLDSYS(*reason-id*)) operand.

- A **user reason ID** is defined by the user.

Resolution of user reason IDs depends on why you specified the user reason ID and on the scope of the problem.

- A **hold class** is a 1- to 7-character value identifying an alternative way to resolve another hold type. For example, a PTF may be held for an error reason ID, but IBM has determined that the problem it resolves is more critical than the error reflected by the holding APAR. The ++HOLD statement for this PTF would have an error reason ID of the holding APAR and a class value of ERREL. If you decide to install the PTF, you can specify **BYPASS(HOLDCLASS(ERREL))** and not have to resolve the error reason ID.

Class values can be resolved **only** with the BYPASS operand.

## SYSMOD Installation

After determining which SYSMODs are to be applied, SMP/E performs the following tasks to install the SYSMODs:

1. Determine the order in which the SYSMODs should be processed.
2. Perform delete processing for any SYSMODs with the DELETE operand specified on their ++VER statement.
3. Move, delete, or rename specified elements or load modules.

4. Process any inline JCLIN.
5. Select elements from the SYSMODs to be applied.  
**Note:** The previous three items are combined and are performed as each SYSMOD is processed.
6. Call system utilities to install the selected elements.
7. Update the applicable target zone entries.
8. Produce summary reports identifying all completed processing.

The following sections describe each of these tasks in greater detail.

### SYSMOD Processing Order

SMP/E orders the processing of the set of SYSMODs being applied to ensure proper processing of JCLIN, element selection, and merges of source and macro updates.

The order in which the SYSMODs being applied are processed is determined from the prerequisite (PRE) data supplied on the ++VER statement for the SYSMOD. SYSMODs named as prerequisites are processed before the SYSMODs naming them.

If no prerequisite order can be determined between SYSMODs, function SYSMODs are processed first, followed by service SYSMODs (PTFs, then APARs, then USERMODs).

### Deleted SYSMODs

A function SYSMOD can delete another function by naming the function to be deleted as an operand of the ++VER DELETE operand. SMP/E deletes that function and all functions, PTFs, APARs, and USERMODs dependent on it. The functions specifically named in the DELETE operand list are considered *explicitly* deleted SYSMODs; all SYSMODs deleted because of their dependence on the explicitly deleted SYSMODs are termed *implicitly* deleted SYSMODs.

When one function SYSMOD deletes another, SMP/E attempts to remove from the target zone all information related to the deleted SYSMOD. In addition, SMP/E removes from the target libraries all elements currently owned by the deleted function SYSMOD. The following processing is done:

1. SMP/E determines whether there are any function SYSMODs in the hierarchy of the function SYSMOD being deleted, and considers those function SYSMODs to also be eligible for delete processing.
2. SMP/E deletes any SYSMOD having the same FMID value as one of the function SYSMODs being deleted.
3. SMP/E determines all the elements that are currently owned by one of the function SYSMODs to be deleted.
4. SMP/E deletes from the target libraries all the elements of the SYSMODs to be deleted. After the elements are successfully deleted, SMP/E deletes the element entries from the target zone.

If a module is being deleted, SMP/E checks whether the module is contained in any cross-zone load modules. If so, SMP/E deletes the contents of the MOD

entry, except the XZLMOD subentries. If the module is not reintroduced and the AUTOMATIC option is in effect for the cross-zone, the module is deleted from the cross-zone load module during cross-zone processing. For more information, see “TARGETZONE Entry (Target Zone)” on page 782.

5. For load modules composed entirely of modules that are to be deleted, SMP/E deletes the load modules and any aliases for the load modules from the target libraries. If the load modules were successfully deleted, SMP/E deletes the MOD and LMOD entries from the target zone. The load modules are also deleted from the SMPLTS, if applicable.

**Note:** If all the modules in a load module are being deleted or replaced, SMP/E checks whether the load module contains cross-zone modules. If so, SMP/E does **not** delete the load module. Instead, it removes the deleted modules from the load module (leaving a *stub* load module in the target libraries) and leaves the LMOD entry in the target zone.

6. For load modules composed of modules to be deleted and modules not to be deleted, SMP/E delinks the CSECTs in the deleted modules from the load module. In priority order, SMP/E obtains the CSECT information in the following manner:
  - a. By gathering the list of CSECT names in the target zone MOD entry
  - b. By determining the CSECT operand on the ++MOD statement from the lowest function or service level SYSMOD being installed that contains that module
  - c. By assuming that the CSECT name is equal to the distribution library name

The MOD entries are deleted from the target zone, but the LMOD entry remains because it is still needed to process SYSMODs affecting the modules that have not been deleted.

For each deleted module, SMP/E adds to the LMOD entry a MODDEL subentry for the module name. MODDEL subentries document the connection between the deleted modules and the LMOD. If any of these deleted modules are ever reintroduced, SMP/E looks for LMODs with MODDEL subentries for the modules and automatically rebuilds the LMODs to include these modules again. The MODDEL subentries for the reintroduced modules are then removed from the LMOD entries.

Any aliases associated with the load modules are not deleted from the target libraries, but may be marked nonexecutable by the link-edit utility.

7. The target zone SYSMOD entries for all implicitly deleted SYSMODs are deleted. For each explicitly deleted SYSMOD, a target zone SYSMOD entry is created. This entry has a DELBY subentry naming the function that caused the deletion. The SYSMOD entries for the explicitly deleted SYSMODs prevent the deleted function SYSMODs from being reprocessed by APPLY.

**Note:** For a function that both deletes and supersedes another function, the SYSMOD entry contains a SUPBY subentry instead of a DELBY subentry. This allows SYSMODs naming the deleted function as a requisite to still be installed.

The result of this process is the deletion of all SYSMODs in the hierarchy of the specified function SYSMOD.



**Note:** Always check the APPLY output to verify that all the CSECTs that were supposed to be deleted from a load module have been deleted.

In Figure 4, function SYSMODs HDE1203, HDE1303, and HDE1403, and service SYSMODs UZ00009, UZ00010, and UZ00004 are deleted because DELETE(HDE1203) is specified on the ++VER statement. CIFREQ subentries in the SYSMOD entry for a function that is deleted (either explicitly or implicitly) are retained in the SYSMOD entry along with the DELBY subentry. So, when function HDE2000 is applied, CIFREQ subentries in the SYSMOD entry for function HDE1203 are retained, as are any CIFREQ subentries in the SYSMOD entries for functions HDE1303 and HDE1403. Likewise, any CIFREQ subentries for conditional requisites specified by the deleted SYSMODs are retained in the appropriate SYSMOD entries.

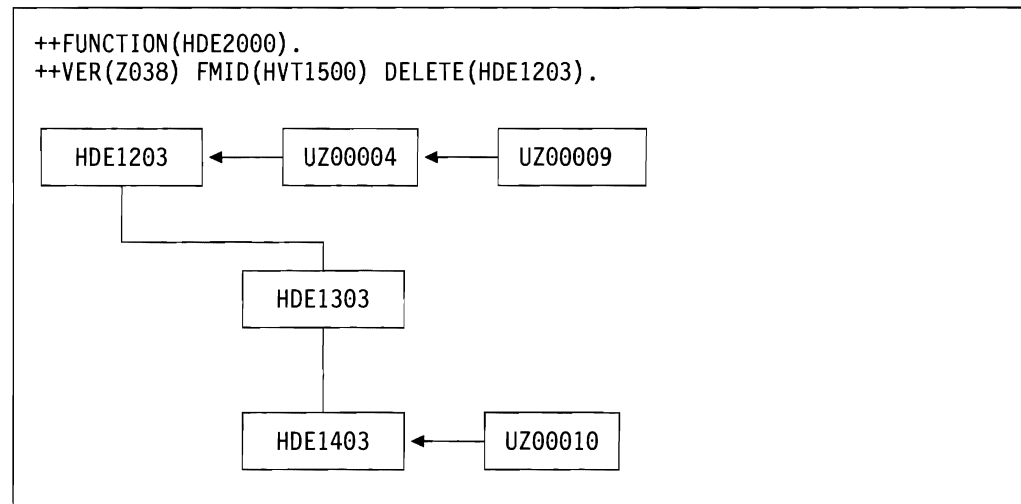


Figure 4. DELETE Hierarchy for DELETE(HDE1203): APPLY Processing

**Note:** Remember, SMP/E assumes that when a function is deleted, the deleting function replaces all the required elements of the deleted function. Although you can build a function SYSMOD that does nothing but delete another function, it is your responsibility to make sure your system remains functionally complete after the product has been deleted.

During APPLY processing, when a function is deleted from a target zone by another function, its FMID is not removed from the FMID list in the global zone. This is because the deleted function can still be applied in other target zones or accepted in other distribution zones.

### Inline JCLIN

Inline JCLIN data for a SYSMOD is supplied following the ++JCLIN statement. JCLIN processing is done before element processing in order to prepare the target zone.

Each entry in the target zone that is affected by the JCLIN update is saved as BACKUP entries on the SMPSCDS before the update. These BACKUP entries record the SYSMOD ID of the SYSMOD that contained the inline JCLIN and the type of update performed.

### Notes:

1. Inline JCLIN is not processed for superseded or deleted SYSMODs.
2. Inline JCLIN does **not** cause SMP/E to update the target libraries; only the entries in the target and distribution zones are updated. These libraries are updated when SMP/E processes the elements in the SYSMOD. The element statements in the SYSMOD determine which elements should be installed.
3. If SMP/E is creating an LMOD entry, and if it finds an existing LMOD entry that is for the same load module and that contains only cross-zone subentries:
  - SMP/E issues messages indicating that the cross-zone relationship might no longer be valid, and then deletes the cross-zone subentries from the load module.
  - If deleting these cross-zone entries eliminates a TIEDTO relationship with a cross-zone, SMP/E deletes the associated TIEDTO value from the TARGETZONE entry for the set-to zone. For an explanation of the TIEDTO value, see “TARGETZONE Entry (Target Zone)” on page 782.
  - Entries for related cross-zone modules are **not** updated to indicate that they are no longer part of the load module in the set-to zone.
  - You must determine whether the cross-zone relationship is still valid. If so, reestablish it by using the LINK command. For more information about the LINK command, see Chapter 9.

The NOJCLIN operand on the APPLY command prevents the processing of inline JCLIN. NOJCLIN can be used if the JCLIN contains data that would overlay user-modified entries in the target zone.

If you specify **NOJCLIN** without an operand list, inline JCLIN is not processed for any SYSMOD that was selected and that contained a ++JCLIN statement. If you specify **NOJCLIN** with an operand list, inline JCLIN is not processed for the specified SYSMODs.

For more information about JCLIN processing, see Chapter 8.

### Moving, Deleting, and Renaming Elements and Load Modules

Macros, modules, source, and load modules can be moved from one target library to another by use of the ++MOVE statement. In addition, load modules can be deleted from a target library by use of the ++DELETE statement, or renamed by use of the ++RENAME statement. This processing is done before element processing.

When ++DELETE is processed, the load module is deleted from the target libraries. If the LMOD entry had a CALLLIBS subentry list, the load module is deleted from the SMPLTS library.

When ++RENAME is processed, the load module is renamed in the target libraries. If the LMOD entry had a CALLLIBS subentry list, the load module is renamed in the SMPLTS library.

Each entry in the target zone that is moved or renamed is saved in a BACKUP entry on the SMPSCDS before the update is performed. Each BACKUP entry in

the SMPSCDS records the SYSMOD ID of the SYSMOD that contained the ++MOVE or ++RENAME statement and the type of update performed.

**Note:** No BACKUP entries are created when a load module is deleted.

**Cross-Zone Load Modules:** If a SYSMOD being applied contains a ++RENAME statement for a load module containing cross-zone modules, SMP/E checks whether those zones indicate that cross-zone updates should be done automatically. If so, the cross-zone MOD entries are updated during cross-zone processing. If a SYSMOD being applied contains a ++DELETE statement for a load module containing cross-zone modules, SMP/E deletes the load module, as well as all the contents of the LMOD entry except for the XZMOD and XZMODP sub-entries.

The ++MOVE, ++DELETE, and ++RENAME statements are further described in Chapter 32.

## Element Selection

SMP/E uses the element statements provided in a SYSMOD to determine which elements should be installed in the target libraries. The selection of elements from a SYSMOD is based on relationships among SYSMODs being installed, other SYSMODs being installed, and modification identifiers of the corresponding elements installed on the target system. Three modification identifiers are kept for each element:

- FMID: Function modification identifier

The FMID of an element is the function SYSMOD that owns the element. Generally, an element's FMID is established (and later changed) by the installation of a function SYSMOD. In this case, the element's FMID is the function SYSMOD that installed the element on the target system.

- RMID: Replacement modification identifier

The RMID of an element is the last SYSMOD that replaced the element (or caused the element's FMID to change). An element's RMID is established by the SYSMOD that first introduces the element to the target system. The RMID of an element is changed by the installation of a SYSMOD that supplies a replacement for the element. Element replacements are ++MOD, ++MAC, ++SRC, and data element MCSs, and modules resulting from assemblies.

- UMID: Update modification identifier

The UMIDs of an element are the set of SYSMODs that have applied updates to the target system element. A UMID is added to the set of the element's UMIDs for each SYSMOD that applies an update to the element. Whenever a new replacement for the element is applied, the set of UMIDs is cleared to start anew with subsequent updates applied to the new replacement. Element updates are ++ZAPs, ++MACUPDs, and ++SRCUPDs.

**Note:** Because data elements can only be replaced and cannot be updated, they do not have UMIDs.

The purpose of element selection is to make sure that the correct functional level of each element is selected and that no service is inadvertently removed from the system.

Element selection in SMP/E is divided into three cases:

- The FMID of the SYSMOD being installed matches the FMID of the element on the target system.
- The FMID of the SYSMOD being installed differs from the FMID of the element on the target system.
- A function SYSMOD is being reinstalled.

The following sections describe processing for each case.

### **FMIDs Match: MODID Verification**

In this case, SMP/E is dealing with elements belonging to the same function, and element processing is based on service relationships expressed by means of the PRE and SUP operands.

The following checks are made for the elements in a SYSMOD to ensure that a proper relationship exists between the SYSMOD being installed and previously installed SYSMODs that supplied the same elements.

**All Elements:** The SYSMOD being installed must specify the RMID of the associated target system element on the PRE or SUP operand. If the RMID of the target system element is the same as its FMID, the element has not been replaced by any SYSMOD, and so the SYSMOD being installed need not specify the RMID value in the PRE or SUP operand.

If the element being installed is a ++SRC/++SRCUPD or a ++MAC/++MACUPD resulting in an assembly that replaces a target system module (element), the SYSMOD being installed must specify as one of its PRE or SUP operand values the RMID of the corresponding target system module that is replaced by the assembly. If the target system module is itself the result of an assembly (RMIDASM indicator set in the MOD entry), an exception to this requirement is made, because the reassembly picks up any changes caused by the SYSMOD that last replaced the module through an assembly.

If the SYSMOD being processed does not specify the RMID of the element on the PRE or SUP operand, SMP/E does not apply that SYSMOD, because doing so would regress the service supplied by the SYSMOD represented by the RMID. If you want to allow the RMID SYSMOD to be regressed, you can specify the BYPASS(ID) operand of the APPLY command.

**Replacement Elements:** The SYSMOD being installed must be a prerequisite for, or must supersede, all UMIDs associated with the target system element.

As above, assemblies resulting from ++SRC/++SRCUPD and ++MAC/++MACUPD elements are considered replacement modules; the SYSMOD being installed must specify on the PRE or the SUP operand all UMIDs of the corresponding target system modules that are replaced by the assembly. No exception is made for a SYSMOD that does not specify on the PRE or the SUP operand all UMIDs associated with modules that have been assembled, because any UMIDs associated with the module are ZAPs that are overlaid by a new assembly.

If the SYSMOD being processed does not specify the UMID values of the elements on the PRE or the SUP operand, SMP/E does not apply that SYSMOD, if this

SYSMOD were to be applied, each of the SYSMODs represented by those not specified in either the SUP or PRE operands would be regressed. To allow the regression, use the BYPASS(ID) operand on the APPLY command. The MODID check condition is then reported as a warning, and the elements are installed on the target libraries.

**Update Elements:** It is assumed that previous updates are still there and will be incorporated with the current update. Therefore, a SYSMOD need not state a relationship (PRE or SUP) to a previous update.

The SYSMOD being installed need not specify on the PRE or SUP operand the UMIDs associated with the corresponding target system element. If any element UMIDs in the target system are not specified in the SUP or PRE operands, a MODID check warning condition is raised and is reported to the user.

The MODID check warning does not result in the termination of the SYSMOD being installed, and the update is installed on the target system. The warning is given because SMP/E is unable to determine with certainty that the two modifications in fact have a relationship or that there is an intersection. Thus, it is the responsibility of the developer or the service team (that is, whoever supplies the update type SYSMOD) to make sure that this SYSMOD specifies the correct relationships with all previous SYSMODs.

### FMIDs Differ

In this case, SMP/E is dealing with elements belonging to different functions, and element selection is based on functional relationships expressed via FMID and VERSION. Elements can be excluded (that is, not selected), and processing of the SYSMOD continues under the assumption that a functionally higher version of the element is already installed on the target system.

An element is **excluded** from the SYSMOD being installed unless one of the following conditions is met:

- The function SYSMOD being installed names the FMID of the target system element in the ++VER FMID operand. In this case, the function being installed is superior to the function that owns the target system element; therefore, the element is selected.
- The MCS associated with the element from the SYSMOD being installed has a VERSION operand, and the FMID of the target system element is named in the VERSION list. In this case, the element from the SYSMOD being installed is considered functionally superior to the target system element, and it is selected.

If there is no VERSION operand on the MCS of an element, the SYSMOD IDs named in the VERSION operand on the ++VER are used as described above. In this situation, SMP/E may be dealing either with a function SYSMOD or with a nonfunction SYSMOD that is changing the functional ownership (FMID) of the elements.

**Note:** If a SYSMOD containing an element update (++SRCUPD, ++MACUPD, or ++ZAP) attempts to change the ownership (FMID) of the element (via the VERSION operand), the SYSMOD cannot be installed.

When an element is selected, its FMID becomes that of the SYSMOD from which it is selected. No further MODID checking is done for these elements; SYSMODs are

constructed so that when the functional ownership of a module changes, either the SYSMOD changing the ownership or the data stored in the target zone SYSMOD entries (the CIFREQ data) contains sufficient information to prevent any service or functional regressions.

### Reinstalling a Function

Element selection gets more complicated only for **function** SYSMODs that are being reinstalled and have elements that intersect with corresponding elements having the same FMID as themselves (see “FMIDs Match: MODID Verification” on page 88). The processing for this situation proceeds as in “FMIDs Match: MODID Verification” on page 88. When a MODID check error condition is detected, however, SMP/E checks further to determine whether the service level of the target system element is higher than that of the element from the SYSMOD being reinstalled. If so, the element from the SYSMOD being reinstalled is not selected, and processing of the SYSMOD continues. If not, the SYSMOD is terminated with a MODID check error.

### APPLY CHECK Processing

If the CHECK operand of APPLY was specified, processing stops at this point. SMP/E produces all the normal APPLY reports, assuming that any SYSMOD not already reported as having a problem will be successful during the real APPLY run. These reports can be used to determine the following:

- Missing DD statements
- Missing requisite SYSMODs
- Regressions
- SYSMOD in hold exception status
- Processing of inline JCLIN

## Building Load Modules

After selecting the elements to be installed, SMP/E verifies that all the modules needed to build any new load modules are available. Some new load modules are made of modules from previously installed SYSMODs, as well as from new modules. If a module is not contained in the SYSMOD being processed, SMP/E checks whether the target zone contains a MOD entry for that module. If it finds one, it checks whether the entry points to an existing LMOD entry and looks for the following:

- If there is an LMOD entry and it contains a COPY indicator, SMP/E can include the module from the target library indicated in the LMOD entry.
- If the LMOD entry does not contain a COPY indicator, SMP/E checks whether the load module is a single-CSECT load module. If so, SMP/E can include the load module from the target library indicated in the LMOD entry.
- If the LMOD entry does not contain a COPY indicator and is **not** a single-CSECT load module, SMP/E assumes that the module has been assembled and looks for an ASSEM or SRC entry with the same name as the module. If SMP/E finds an ASSEM entry in the target zone, it assembles the entry and uses the output when it link-edits the load module. If SMP/E finds an SRC entry in the target zone, the SMPSTS data set, or the distribution library, it assembles the source and uses the output when it link-edits the load module.

If SMP/E could not find a useable copy of the module, SMP/E checks whether there is a DLIB entry corresponding to the distribution library for the module. If so, SMP/E checks whether the module is in the target library pointed to by the DLIB entry. If this is true, SMP/E uses that copy of the module when it link-edits the load module.

If a module is missing from the target zone or the target libraries and is not a new module, SMP/E checks for it in the distribution zone. If the FMID, RMID, and UMID in the distribution zone MOD entry match those in the target zone MOD entry, SMP/E uses the distribution library version of the module to build the load module during element installation. If any modules are missing, the load module is built without them.

**Note:** SMP/E checks whether load modules to be updated contain modules from a cross-zone with the same name as modules currently selected to update the load module. This is done by checking the load module's XZMOD sub-entries. If this condition exists, SYSMOD processing stops.

## Element Installation

Once the proper SYSMODs have been selected and the proper functional and service level of each element has been determined, SMP/E begins the process of calling utility programs to get the elements installed in the appropriate target libraries. The following sections describe the process of installing each of the element types supported by SMP/E.

### Deleting Elements

When the DELETE operand is specified on the element MCS, macros, modules, source, data elements, and HFS elements are deleted from a target library. When SMP/E processes an element specifying the DELETE operand in its MCS statement, it first saves the current element entry from the target zone in a BACKUP entry on the SMPSCDS. This BACKUP entry includes the SYSMOD ID of the SYSMOD containing the MCS statement that caused the change, plus an indicator for the type of change (DEL) made. After SMP/E saves the BACKUP entry for the element, it deletes the element from the target library and the target zone.

**Note:** If a module is being deleted, SMP/E also checks whether the module is contained in any cross-zone load modules. If so, SMP/E deletes the contents of the MOD entry except the XZLMOD subentries. If the module is not reintroduced and the AUTOMATIC option is in effect for the cross-zone, it is deleted from the cross-zone load module during cross-zone processing. For more information, see "TARGETZONE Entry (Target Zone)" on page 782.

### Compressing the Target Libraries

You can use the COMPRESS operand of the APPLY command to have SMP/E compress the target libraries before installing SYSMODs. There are two methods of specifying which libraries are to be compressed:

- You can specify a list of specific libraries in the COMPRESS operand (including libraries that may not be affected by the APPLY command).
- You can specify ALL on the COMPRESS operand, in which case the only libraries eligible for compression are those in which elements will be installed by this APPLY command.

**Note:** Target libraries residing in a hierarchical file system are not compressed.

Once the libraries have been determined, actual processing is dependent on the library type:

- For **source** libraries, any source to be replaced (not updated) by one of the SYSMODs being installed is deleted from the library.
- For **macro** libraries, no deleting is done. This is because the SYSMOD replacing the macro might fail, and there may be other SYSMODs that cause assemblies that use the macro.
- For **data element** libraries, any data element to be replaced by one of the SYSMODs being installed is deleted from the library.
- For **load** libraries, SMP/E determines which load modules within the library are composed only of modules copied during system generation and are being replaced by the SYSMODs being installed. Such load modules are deleted from the library. If a load module contains a module that is not being replaced, that load module is not deleted.

SMP/E then calls the copy utility to perform the actual compress operation.

**Notes:**

1. To reclaim as much space as possible before installing the SYSMODs, members are deleted before the library is compressed. SMP/E processes one library at a time, first deleting members, then compressing the library.
2. When you install a function that deletes another function but supplies no elements, no libraries are compressed.

### Macro Replacements

One of the steps in actually updating the target libraries is to install macro replacements. Multiple SYSMODs can be applied, each of which can contain a replacement for the same macro. When two or more SYSMODs replacing the same macro are applied concurrently, SMP/E determines the version at the highest function and service level. For a full explanation of how SMP/E determines the functional and service level of an element, see “Element Selection” on page 87.

SMP/E then schedules the actual update of the target macro library. The library to be updated is determined from the SYSLIB information in the target zone MAC entry. For further information about the initial setting of the MAC SYSLIB, see “Adding New Elements Other Than Modules to the Target Libraries” on page 67.

If there is no SYSLIB value in the MAC entry, SMP/E looks for a DLIB entry with the same name as the DISTLIB value in the MAC entry and uses the SYSLIB value from the DLIB entry.

**Note:** In this case, before you run the APPLY command, make sure there is only one SYSLIB subentry in the DLIB entry and that it specifies the ddname of the correct library.

If no SYSLIB is present, the SMPMTS is used as the target macro library. (For further information about the use of the SMPMTS as a target macro library, see “Use of the SMPMTS and SMPSTS as Target Libraries” on page 70.)



The actual update is done by calling either the copy utility or the update utility.

- The copy utility is used in these cases:
  - The macro was packaged in a relative file (the RELFILE operand was specified).
  - The macro was packaged in a text library (the TXLIB operand was specified) and had no alias names—that is, no MALIAS names are in the target zone MAC entry and no MALIAS operands are on the ++MAC statement.
  - The macro was packaged inline, it had no alias name, and the SSI operand was not specified.

The SSI operand is ignored when TXLIB or RELFILE is specified.

- The update utility is called in these cases:
  - The macro was packaged in a text library, and the element had an alias name.
  - The macro was packaged inline, and either it had an alias name or the SSI operand was specified.

Upon return from either utility, SMP/E issues a message indicating whether the macro was replaced successfully.

### Macro Updates

After all the macro replacements have been done, any macro updates present can be scheduled. Again, multiple SYSMODs may contain updates for the same macro. When two or more updates to the same macro are being processed concurrently, SMP/E merges the text from each update based on the sequence numbers in columns 73 to 80. The order of the merge is based on the processing order expressed by the PRE operands on the ++VER statements or by internal defaults, or both, as follows:

- If SMP/E finds a processing order relationship between all the SYSMODs being processed, the merge occurs according to that order.
- If any of the SYSMODs being processed do not have a processing order relationship with other SYSMODs that do have a processing order, the updates from the unrelated SYSMODs are merged after the updates from SYSMODs that have a processing order relationship.
- If SMP/E cannot determine the processing order of the SYSMODs, it merges the updates for PTFs first, APAR fixes second, and USERMODs third. Within each type, there is no specified order.

SMP/E then calls the update utility to perform the actual target library updating.

The library to be updated is determined from the SYSLIB information in the target zone MAC entry. For more information about how SMP/E determines the MAC SYSLIB, see “Macro Replacements” on page 92. If no SYSLIB is present, the SMPMTS is used as the target macro library. For further information about the use of the SMPMTS as a target macro library, see “Use of the SMPMTS and SMPSTS as Target Libraries” on page 70. If there is no SYSLIB and the macro does not exist in the SMPMTS, the macro is obtained from the distribution library and then updated. Upon return from the update utility, SMP/E issues a message indicating whether the update was successful.

**Note:** SMP/E checks only whether the NAME operand on the ./ CHANGE statement specifies the same element as the ++MACUPD statement. (This checking is done during RECEIVE processing.) Other ./ CHANGE operands may produce undesired results. For example, if you code UPDATE=INPLACE and there is no SYSLIB in the MAC entry, the distribution library may be updated.

### Source Replacements

Another step in actually updating the target libraries is the installation of source replacements. Multiple SYSMODs can be applied, each of which can contain a replacement for the same source.

When two or more SYSMODs replacing the same source are applied concurrently, SMP/E determines the version at the highest function and service level. For full explanation of how SMP/E determines the functional and service level of an element, see “Element Selection” on page 87.

SMP/E then schedules the actual update of the target source library. The library to be updated is determined from the SYSLIB information in the target zone SRC entry. For further information about the initial setting of the SRC SYSLIB, see “Adding New Elements Other Than Modules to the Target Libraries” on page 67.

If there is no SYSLIB value in the SRC entry, SMP/E looks for a DLIB entry with the same name as the DISTLIB value in the SRC entry and uses the SYSLIB value from that DLIB entry.

**Note:** In this case, before you run the APPLY command, make sure there is only one SYSLIB subentry in the DLIB entry and that it specifies the ddname of the correct target library.

If no SYSLIB is present, the SMPSTS is used as the target source library. For further information about the use of the SMPSTS as a target source library, see “Use of the SMPMTS and SMPSTS as Target Libraries” on page 70.

The actual update is done by calling either the update utility or the copy utility.

- The copy utility is used if the source replacement resides in either a text library (that is, the TXLIB operand was specified) or in a RELFILE (the RELFILE operand was specified). When TXLIB or RELFILE is specified, the SSI operand is ignored.
- The update utility is called if the source replacement was contained inline and the SSI operand was specified.

Upon return from either utility, SMP/E issues a message indicating whether the source has been replaced successfully.

### Source Updates

After all the source replacements have been done, any source updates present can be scheduled. Again, multiple SYSMODs can contain updates for the same source. When two or more updates to the same source are being processed concurrently, the text from each is merged. Looking at the sequence numbers in columns 73 to 80, SMP/E processes the updates in the order expressed by the PRE operands on the ++VER statements or by internal defaults or both as follows:

- If SMP/E finds a processing order relationship between all the SYSMODs being processed, the merge occurs according to that order.
- If any of the SYSMODs being processed do not have a processing order relationship with other SYSMODs that do have a processing order, the updates from the unrelated SYSMODs are merged after the updates from SYSMODs that have a processing order relationship.
- If SMP/E cannot determine the processing order of the SYSMODs, it merges the updates for PTFs first, APARs second, and USERMODs third. Within each type, there is no specified order.

SMP/E then calls the update utility to do the actual updating of the target library. The library to be updated is identified from the SYSLIB information in the target zone SRC entry. For more information about how SMP/E determines the SRC SYSLIB, see “Source Replacements” on page 94. If no SYSLIB is present, the SMPSTS is used as the target source library. For further information about the use of the SMPSTS as a target source library, see “Use of the SMPSTS and SMPSTS as Target Libraries” on page 70. If there is no SYSLIB and the source does not exist in the SMPSTS, the source is obtained from the distribution library and then updated. Upon return from the update utility SMP/E issues a message indicating whether the update was successful.

**Note:** SMP/E checks only whether the NAME operand on the ./ CHANGE statement specifies the same element as the ++SRCUPD statement. Other ./ CHANGE operands may produce undesired results. For example, if UPDATE=INPLACE is coded and there is no SYSLIB in the SRC entry, the distribution library may be updated.

## Assemblies

This section describes the following:

- Assembling source
- Assemblies caused by macros
- Reusing previous assemblies

**Assembling Source:** When a SYSMOD contains source to be assembled, it can supply either just the source (++SRC/++SRCUPD) for that element, or it can supply both the source (++SRC/++SRCUPD) for the element and an object deck (++MOD) for the same element.

- **Source only:** When the SYSMOD supplies only the source and no corresponding object deck, the element is assembled.
- **Source and ++MOD:** When the SYSMOD supplies both the source and the corresponding object deck, SMP/E determines whether the object deck can simply be link-edited into the target system or whether the source must be assembled. It does so by considering the following questions:
  - Was ASSEM specified on the APPLY command?
  - Are there any updates to the source that the SYSMOD supplying the object does not know about (UMIDs in the target zone SRC entry)?
  - Has another SYSMOD, which this SYSMOD does not know about, assembled the module (RMID of the target zone MOD entry for the module to be assembled)?

- Is the ASSEMBLE indicator set for the corresponding MOD?

If the answer to any of the above questions is yes, the module is assembled if SMP/E can determine where the resultant assembled object should go. SMP/E knows where to put the assembled object if there is a target zone MOD entry for the resultant object and a load module into which the module should be link-edited. If the MOD entry cannot be found or is not included in the same SYSMOD, a warning message is issued. Processing of the SYSMOD continues without assembling or link-editing the module. For further information on how SMP/E processes the object deck after assembly, see “Module Replacements” on page 97.

**Assemblies Caused by Macros:** A SYSMOD can supply macros that require the assembly of modules. The required assemblies are found as GENASM subentries in the target zone MAC entry and in the ASSEM and PREFIX operands on the ++MAC/++MACUPD statement. The source for these assemblies is found by looking for a target zone ASSEM entry matching the name of the module; if an ASSEM entry is not found, the SRC entry matching the name of the module is used as the source. If neither an ASSEM nor a SRC entry can be found, a warning message is issued, and SMP/E goes on processing the SYSMOD without assembling or link-editing the module.

The SYSMOD can also supply an object deck for the modules to be assembled. SMP/E determines whether to do the assembly rather than use the object decks supplied in the SYSMOD. It makes this determination by considering the following questions:

- Was ASSEM specified on the APPLY command?
- Are there any updates to the macro that the SYSMOD supplying the object does not know about (UMIDs in the target zone macro entry)?
- Has another SYSMOD, which this SYSMOD does not know about, assembled the module (RMID of the target zone MOD entry for the module to be assembled)?
- Is the ASSEMBLE indicator set for the corresponding MOD?

If the answer to any of the above questions is yes, SMP/E assembles the module if it can determine where the resultant assembled object should go. SMP/E knows where to put the assembled object if there is a target zone MOD entry for the resultant object and a load module into which the module should be link-edited. (See “Module Replacements” on page 97 for further information.)

Whenever a macro modification in a APAR or USERMOD type SYSMOD causes the assembly of a module, the ASSEMBLE indicator in the module's target zone MOD entry is set on. If any subsequent PTF, APAR, or USERMOD type SYSMODs contain modifications to macro or source elements affecting a module whose ASSEMBLE indicator has been set, they cause the module to be reassembled in spite of the presence of an object module in the SYSMOD. The reassembly prevents the assembled module from regressing during the installation of subsequent SYSMODs that might replace the affected module but that do not contain the macro modification introduced by the earlier SYSMOD.

To prevent future reassemblies of modules that have had their ASSEMBLE indicator set, the UCLIN function must be used to set it off (DEL).

**Reusing Previous Assemblies:** If SMP/E is run after a failure, assemblies are rerun to ensure that the proper source and macros are used. If the same set of SYSMODs is being processed after a failure, the assemblies run before the failure need not be rerun. The previously assembled objects for the failed SYSMOD are used if the REUSE operand is specified on the APPLY command.

Assembled object decks are stored on the SMPWRK3 data set. If SMPWRK3 is allocated with a final disposition of KEEP, these assembled modules are saved in SMPWRK3 until the SYSMODs causing the assemblies are successfully processed. This allows SMP/E to reuse the assembled object modules if the APPLY command fails. Once the SYSMODs have been successfully applied, SMP/E deletes the related entries from SMPWRK3 data set.

**Note:** SMP/E does not check to make sure the same set of SYSMODs are being rerun after a failure. The user must proceed with care in taking advantage of the REUSE facility.

### Module Replacements

The modules (++MODs and assemblies from ++SRCs) selected from a SYSMOD are generally link-edited to a load module library on the target system. SMP/E determines the load module to which a module belongs by checking for load module (LMOD) subentries in the target zone MOD entry for the module and by looking at the MODDEL subentries for all LMOD entries. The link-edit characteristics and control statements for the link-edit are found in the target zone LMOD entry for the appropriate load module. For details on information that is passed to the link-edit utility, see “Link-Edit Parameters and Load Module Attributes.”

**Note:** If SMP/E cannot determine the load module for a module, it presumes that the configuration of the target system does not require the module, and does no link-edit or copy. However, it replaces the RMID subentry of the MOD entry with the ID of the SYSMOD supplying the ++MOD or causing the assembly.

Specific processing depends on whether the load module was part of a totally copied library, selectively copied, or defined by link-edit JCL. In addition, special processing is done for the nucleus load module (IEANUC01). For more information, see these sections:

- “Load Modules in Totally Copied Libraries” on page 98
- “Load Modules That Were Selectively Copied” on page 98
- “Load Modules Defined by Link-Edit JCL” on page 98
- “Processing for the Nucleus Load Module (IEANUC01)” on page 100

**Link-Edit Parameters and Load Module Attributes:** The parameters passed to the link-edit utility include the default link-edit **parameters** (LET, LIST, and XREF) and load module **attributes** (such as RENT, REUS, REFR, AMODE=24). SMP/E determines the attributes and parameters to be passed as follows:

- If SMP/E has determined that the binder is available on the system, the STORENX link-edit parameter is passed to the link-edit utility.
- If the load module was link-edited during initial installation, the link-edit parameters saved in the LMOD entry are used.

- If the load module was copied during initial installation (that is, the LMOD COPY indicator is on):
  - If **LEPARM** is specified on the ++MOD statement, the attributes supplied are used, and the corresponding target zone LMOD entry is updated (or created) with these attributes.
  - If **LEPARM** is not specified and a target zone LMOD entry exists, and the LMOD entry has the link-edit attributes stored in it, the attributes from the LMOD entry are used.
  - If **LEPARM** is not specified and a target zone LMOD entry does not exist (or exists without link-edit attribute indicators), the target library is accessed to determine the link-edit attributes for the load module. If the load module is not found on the target library, no load module attributes are passed to the link-edit utility unless the user has set some in the UTILITY entry for link-edit processing.
- If a link-edit UTILITY entry is in effect, the defaults are not passed to the link-edit utility. The values in the UTILITY entry are used instead.
- Regardless of whether a UTILITY entry is in effect, SMP/E determines whether to pass the NCAL or CALL parameter based on whether the LMOD entry for the load module contains a CALLLIBS subentry.

**Load Modules in Totally Copied Libraries:** SMP/E checks to determine whether the distribution library (DLIB) for the module has been totally copied to a target system library. (SMP/E makes this determination by looking for a target zone DLIB entry describing the module's DLIB.) If this is the case, SMP/E creates a load module on the target library having the same name as the module's name, if one does not already exist. An INCLUDE statement is generated for the module, but no INCLUDE statement is generated for the current version of the load module.

**Load Modules That Were Selectively Copied:** If the LMOD entry for the load module contains the COPY indicator, SMP/E knows that the load module was created by copy JCL and that the load module contains only the one module. If the module is supplied on a LKLIB or relative file, SMP/E copies the module to the target system. If aliases are specified for such a module (as indicated by TALIAS values in the MOD entry), they must exist in the LKLIB or relative file in order to be copied. An INCLUDE statement is generated for the module, but no INCLUDE statement is generated for the current version of the load module.

**Load Modules Defined by Link-Edit JCL:** When SMP/E links a module into a load module that was defined by link-edit JCL, the link-edit utility INCLUDE statements that are generated depend on whether the load module currently exists in a target library. (Processing is different when the load module is defined with a SYSLIB allocation—that is, its LMOD entry has a CALLLIBS subentry list. For details, see “Building Load Modules with a SYSLIB Allocation” on page 99.)

- If the load module does not exist, an INCLUDE statement is built for each module defined in the link-edit JCLIN as being included in the load module. No INCLUDE statement is built for the load module itself.
- If the load module does exist, an INCLUDE statement is built for each module that is selected from a SYSMOD being processed and that is defined in the link-edit JCLIN as being included in the load module. In addition, an INCLUDE statement is built to include the current version of the load module from the

target library. This is done to obtain the other modules that make up the load module.

Link-edit control statements saved in the target zone LMOD entry are passed to the link-edit utility as SYSLIN input.

*Building Load Modules with a SYSLIB Allocation:* For each load module to be built, SMP/E determines whether a SYSLIB allocation is required. It does this by checking the corresponding LMOD entry for a CALLLIBS subentry list. If a SYSLIB allocation is required, the allocation is done before the load module is link-edited. Each ddname in the CALLLIBS subentry list is dynamically allocated using information from the corresponding DDDEF entry, and is assigned an SMP/E-generated ddname. When the CALLLIBS subentry list contains more than one name, SMP/E allocates them as a concatenation and gives it a generated ddname. If errors occur during the SYSLIB allocation, dynamic allocation error messages are issued, and the load module is not link-edited. Any SYSMODs supplying modules for the link-edit are also failed.

The load module is built in two stages: first, a “base” version of the load module is built in the SMPLTS data set; second, the executable version of the load module is built in the target libraries.

1. **Building the “base” version of the load module.** The appropriate INCLUDE statements are built:

- If the load module does **not** currently exist in the SMPLTS library, an INCLUDE statement is built for each module explicitly defined in the link-edit JCLIN as being included in the load module. No INCLUDE statement is built for the load module itself.

**Note:** This is done even if the executable version of the load module exists in the target libraries.

- If the load module **does** currently exist in the SMPLTS data set, an INCLUDE statement is built for each module that is selected from a SYSMOD being processed and that is explicitly defined in the link-edit JCLIN as being included in the load module. In addition, an INCLUDE statement is built to include the current version of the load module from the SMPLTS library. This is done to obtain the other modules that make up the load module.

When the “base” version of a load module is link-edited into the SMPLTS, all other link-edit control statements defined for the load module (except ALIAS statements) are passed to the link-edit utility, as well as all link-edit options defined for the load module. This link-edit results in unresolved external references, which is considered normal.

2. **Building the executable version of the load module.** The executable version of the load module is built in the target libraries using the load module's SYSLIB allocation (the CALLLIBS subentry list in its LMOD entry) and the “base” version of the load module from the SMPLTS data set. The only INCLUDE statement built is for the “base” version of the load module from the SMPLTS data set.

### Notes:

- a. If the “base” version of the load module does not exist in the SMPLTS data set, the load module is not link-edited.
- b. A load module can reside in an executable target library before a base version of it has been built in the SMPLTS. If the load module had included cross-zone modules through the use of the LINK command, these modules are no longer included after the installation of a SYSMOD that causes the load module to be built into the SMPLTS. (Warning messages are issued to indicate this.) After the installation of such a SYSMOD, the LINK command needs to be rerun in order to include those cross-zone modules back into the load module.

**Processing for the Nucleus Load Module (IEANUC01):** SMP/E recognizes IEANUC01 as a special load module, and, therefore, performs special processing whenever IEANUC01 is to be updated.

- **Backup copy.** At the start of APPLY processing, when SMP/E determines that IEANUC01 will be relinked, a backup copy, IEANUC0x, is saved, where x is the NUCID value in the current OPTIONS entry. However, in certain cases, a backup copy is not made or may be lost:
  - If the OPTIONS entry has no NUCID value or if the value is 1, **no** backup copy of the nucleus is created.
  - If two APPLY commands are processed, each of which causes the nucleus to be replaced, and if the NUCID value is not changed between the first and second APPLY, the backup copy from the first APPLY is lost.
  - If a combination of APPLY and LINK commands is processed, each of which causes the nucleus to be replaced, and if the NUCID value is not changed between the two commands, the backup copy from the first command is lost.

Make sure your nucleus data set is large enough to contain as many copies of the nucleus as required (at least three). The backup copy of the nucleus is not used by SMP/E in order to restore a SYSMOD. It is created so you can use it to IPL an alternative nucleus if you are not able to IPL the system after installing the SYSMODs.

- **Deleted modules.** Unlike for other load modules, MODDEL subentries for IEANUC01 do not cause SMP/E to automatically link modules back into the nucleus. Although a MODDEL subentry is created for IEANUC01 when a module is deleted from the nucleus, this subentry is ignored when the module is reintroduced.

### Module Updates

For each ++ZAP element within a SYSMOD, SMP/E looks for all the load modules that the distribution module was either linked or copied to (similar to the processing done for ++MODs). SMP/E then attempts to install the superzap to each of these load modules. If the load module being zapped contains a CALLLIBS subentry, the SMPLTS version of the load module is also zapped.

In applying the ZAP, SMP/E performs two passes: the first to process the VER control cards, and the second to process the REP control cards. The REP pass is



performed only if the VER pass has been completed successfully for all load modules to be changed.

### Data Element Replacements

Still another step in updating the target libraries is to install replacements for data elements. Multiple SYSMODs can be applied, each of which can contain a replacement for the same data element. When two or more SYSMODs replacing the same data element are applied concurrently, SMP/E determines the version at the highest function and service level. For a full explanation of how SMP/E does this, see “Element Selection” on page 87.

SMP/E then schedules the actual update of the target library. The library to be updated is identified from the SYSLIB information in the target zone element entry.

If there is no SYSLIB value in the data element entry, SMP/E looks for a DLIB entry with the same name as the DISTLIB value in the data element entry and uses the SYSLIB value from the DLIB entry.

**Note:** In this case, before you run the APPLY command, make sure the DLIB entry contains only one SYSLIB subentry specifying the ddname of the correct target library.

The actual update is done by either the copy utility or SMP/E.

- The copy utility updates the library in these cases:
  - The data element was packaged in a relative file (the RELFILE operand was specified).
  - The data element was packaged in a text library (the TXLIB operand was specified).
  - The data element was packaged inline, and was not transformed by GIMDTS.
- SMP/E updates the library if the data element was packaged inline and has been transformed by GIMDTS. All control information is removed, the transformed data is changed back to its original format, and the target library is updated with the element.

At the end of processing, SMP/E issues a message indicating whether the data element has been replaced successfully.

### Hierarchical File System Element Replacements

Before any processing occurs, SMP/E checks to make sure that the hierarchical file system (HFS) copy utility is available (BPXCOPY is the default). If it is not, SMP/E continues APPLY processing. If it is later discovered that an HFS element needs to be processed from a selected SYSMOD, SMP/E terminates that SYSMOD.

When the HFS copy utility is available, an HFS element is processed by that utility. SMP/E checks the linknames of the current HFS element against those of the replacement before invoking the HFS copy utility. If the linknames are the same, SMP/E calls the HFS copy utility to overlay the current HFS element with the new one. If any of the existing linknames are different, SMP/E deletes only those that are not being replaced. SMP/E then calls the HFS copy utility to replace the HFS element and current linknames, and to create any new linknames.

The HFS element can be packaged in relative file format, text library format, or inline. If the HFS element was packaged inline after being transformed, SMP/E retransforms the element back to its original format before invoking the HFS copy utility.

**HFS Copy Utility Parameters and Alternate ddnames:** The parameters and alternate ddnames passed to the HFS copy utility include the following:

- The parameters include any PARM values specified in the HFSCOPY UTILITY entry, plus parameters generated by SMP/E based on either the contents of the HFS element entry (if one exists) or on the ++HFS MCS for the element.

The SMP/E-generated parameters include the name of the element, its installation format (TEXT or BINARY), an identifier for the resulting utility output, and any linknames associated with the element.

**Note:** The maximum total length of the parameters to be passed is X'FFFF' bytes. If the length exceeds this number, SMP/E truncates the parameters at the limit and passes this value to the HFS copy utility.

- The alternate ddnames include any PRINT values specified in the HFSCOPY UTILITY entry, the ddname of the input data set to be used by the utility, and the ddname of the output data set to be used by the utility (the SYSLIB ddname for the element).

## Recording After Completion

Results of processing are recorded in the following entries:

- “Target Zone Element and LMOD Entries”
- “SMPSCDS BACKUP Entries” on page 103
- “Target Zone SYSMOD Entries” on page 104

### Target Zone Element and LMOD Entries

APPLY processing creates, modifies, and may delete target zone element entries.

- Entry update indicator: When an entry is added by a SYSMOD being processed or modified by inline JCLIN, the SYSMOD's SYSMOD ID is placed in the LASTUPD subentry of the target zone element entry.
- ALIAS subentries: The updates to an element's ALIAS subentries are discussed under “Alias Processing” on page 70.
- LMOD subentries: When the LMOD operand is specified on a ++MOD statement, the values in the operand list are added to the target zone MOD entry as LMOD subentries.
- MODID subentries:
  - The FMID subentry is replaced with the FMID of the SYSMOD from which the modification to the element was selected. If the SYSMOD is a function SYSMOD, the FMID is set to the SYSMOD ID of the function itself.
  - The RMID subentry is changed when a replacement element or assembly is applied. The RMID is set to the SYSMOD ID of the SYSMOD supplying the element.

If a MOD entry is being updated as the result of an assembly for a macro or source, the RMID is replaced with the SYSMOD ID of the SYSMOD supplying the ++MAC or ++SRC (and the RMIDASM indicator is set to reflect

this occurrence). The RMIDASM indicator is set for the module, even if the actual assembly was suppressed because the SYSMOD supplied an assembled version of the module. (See “Source Replacements” on page 94 and “Assemblies” on page 95 for further information.)

If the replacement element's MCS specified an RMID for the element (the RMID operand), the specified value is used.

- When a replacement element is applied, all UMID subentries are deleted.

If the replacement element's MCS specified a list of UMIDs for the element (the UMID operand), these UMIDs replace any existing UMIDs for the element.

- UMID subentries are added when updates for the element are applied. The UMIDs are the IDs of the SYSMODs supplying the updates.

If a SYSMOD with an update modification to an element supersedes another SYSMOD with an update modification to the same element, the UMID subentry for the superseded SYSMOD is deleted from the element entry.

- SYSLIB subentry

If a DLIB entry was used to determine the SYSLIB value identifying the target library for an element or load module, that SYSLIB value is added to the corresponding element entry or LMOD entry.

- CSECT names

The CSECT information from the ++MOD statement is saved in the target zone MOD entry. The information saved is determined in the following way:

- If the SYSMOD that the selected version of the module came from contained the CSECT operand, the CSECT names present there are either added to the target zone MOD entry (if no CSECT information was already there) or are used to replace the existing list of CSECT names in the target zone MOD entry.
- If the SYSMOD that the selected version of the module came from did not contain the CSECT operand, SMP/E checks to see if any other SYSMOD applicable to the same FMID was also being applied. If so, and if any of those SYSMODs contained the CSECT operand, the CSECT information from the SYSMOD at the highest service level is used.

### **SMPSCDS BACKUP Entries**

BACKUP entries are created on the SMPSCDS data set associated with the target zone, so RESTORE processing can recover modifications to target zone entries if a SYSMOD is restored.

**Note:** No BACKUP entries are created when a load module is deleted by the ++DELETE statement.

### Target Zone SYSMOD Entries

For each SYSMOD processed, a SYSMOD entry is created in the target zone. If a SYSMOD entry existed previously (as in the case of reapplication of the SYSMOD), the previous entry is replaced. The entry includes data from the applicable ++VER, subentries for each of the elements included in the SYSMOD package, and indicators that are set when ++IF and ++JCLIN are present.

A SYSMOD is considered successfully processed when all its selected elements have been applied to the appropriate system libraries **and** all its requisites have been successfully processed. Because SMP/E processes any number of SYSMODs with elements in common, it is possible that some SYSMODs have elements that need not be installed in a target library; when this is the case, such SYSMODs are not considered successfully processed until the SYSMODs supplying the higher level versions of the corresponding elements are successful.

If the SYSMOD is not successfully processed, an ERROR status indicator is set in the entry.

**Superseded SYSMODs:** When one SYSMOD is superseded by another, SMP/E makes a record of this by adding the name of the superseding SYSMOD to the entry for the superseded SYSMOD.

- When there is only one superseding SYSMOD, its name is saved in the LASTSUP subentry.
- When there are several superseding SYSMODs, the name of the last one is saved in the LASTSUP subentry, and a complete list is saved in the SUPBY subentry.

If the superseded SYSMOD has not been previously applied, its target zone SYSMOD entry contains only the LASTSUP information. Such a SYSMOD entry is called a “dummy entry.” Because the superseded SYSMOD had not been previously applied, SMP/E does not know what type of SYSMOD it was (function, PTF, APAR, or USERMOD).

**Deleted SYSMODs:** When one SYSMOD is deleted by another, SMP/E makes a record of this by adding the name of the deleting SYSMOD to the DELBY subentry of the deleted SYSMOD. If the deleted SYSMOD has not been previously applied, its target zone SYSMOD entry contains only the DELBY information. Such a SYSMOD entry is called a “dummy entry.” Although the deleted SYSMOD had not been previously applied, SMP/E assumes that it was a function SYSMOD, because only function SYSMODs can be explicitly deleted.

**Conditional Requisite Data:** For each SYSMOD named as an FMID in a ++IF statement, a SYSMOD entry is created with CIFREQ subentries representing the conditional requisite requirements.

If the SYSMOD existed previously, the CIFREQ data is simply added to the existing entry; otherwise, a new SYSMOD entry is created to save the CIFREQ data for use if the FMID is installed later.

**Regressed Element Subentries:** The MODID verification checks described earlier may leave elements open to regression. When potential regression of an element is detected, a record for the SYSMOD that previously modified the element is kept by marking the element subentries in the SYSMOD entry as regressed.

## Cross-Zone Processing

If entries for modules or load modules processed by the APPLY command contain cross-zone subentries and the associated cross-zones can be automatically updated, SMP/E does special processing.

**Modules That Are Part of Cross-Zone Load Modules:** For each module replaced or updated, SMP/E checks the target library for a copy of the module that can be used for link-edit processing. A usable copy exists if:

- A module was supplied as a module replacement
- A module exists as a single module load module in a target library
- A module was assembled (the resultant object module exists in the SMPWRK3 data set)

SMP/E then obtains access to the CSIs containing the cross-zones. It checks the cross-zone LMOD entries to make sure that the set-to zone originally supplied the modules to be replaced, updated, or deleted. It also verifies that the cross-zone contains DDDEF entries for the target libraries needed for link-edit processing. Finally, it invokes the link-edit utility for each load module to be updated.

If the cross-zone load module has a SYSLIB allocation (its LMOD entry contains a CALLLIBS subentry list), SMP/E does the processing described in “Building Load Modules with a SYSLIB Allocation” on page 99. The SMPLTS library used is the one specified by the DDDEF entry in the cross-zone. If no “base” version of the cross-zone load module exists in the cross-zone SMPLTS library, the cross-zone load module is not link-edited.

For each module deleted, SMP/E keeps the *stub* MOD entry in the set-to zone in order to preserve the cross-zone information in the XZLMOD subentry. The module itself is deleted from any cross-zone LMODs in which it is contained. SMP/E also keeps the associated XZMOD subentry for the deleted module in the cross-zone LMOD entry.

**Load Modules That Were Renamed:** For each cross-zone module in a renamed load module from the set-to zone, SMP/E changes the XZLMOD subentry in the cross-zone MOD entry to reflect the name of the new load module.

The Cross-Zone Summary report contains a summary of all cross-zone work done, except for cross-zone work caused by renamed LMODs. This is summarized in the MOVE/RENAME/DELETE report.

## Global Zone SYSMOD Entries

The target zone name is added as an APPID subentry in the global zone SYSMOD entry for each successfully processed SYSMOD. The APPID subentries in the global zone, therefore, reflect the target libraries to which each SYSMOD has been applied.

**Note:** The global zone SYSMOD entry is deleted when the SYSMOD is rejected.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of APPLY processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.

2. SYSMOD selection

Global zone	—	Read with shared enqueue.
SMPPTS	—	Read with shared enqueue.
Target zone	—	Update with exclusive enqueue.

3. Element selection

SMPPTS	—	Read with shared enqueue.
Target zone	—	Update with exclusive enqueue.

4. Utility calling

Target zone	—	Update with exclusive enqueue.
-------------	---	--------------------------------

5. Load module build

DLIB zone	—	Read with shared enqueue.
-----------	---	---------------------------

6. Global zone update

Global zone	—	Update with exclusive enqueue.
SMPPTS	—	Update with exclusive enqueue.
Target zone	—	Update with exclusive enqueue.

7. Cross-zone processing

Cross-zones	—	Read with shared enqueue.
Cross-zones	—	Update with exclusive enqueue.
Global zone	—	Read with no enqueue.

8. Termination

All resources are freed.

---

## Chapter 4. The CLEANUP Command

The CLEANUP command deletes entries from the SMPMTS, SMPSTS, and SMPSCDS data sets. This is helpful for:

- APPLY followed by ACCEPT when several target libraries have been created from the same distribution library: When a SYSMOD is accepted into a distribution zone, the entries associated with the SYSMOD are automatically deleted from the SMPMTS, SMPSTS, and SMPSCDS for the related target zone. However, even if the SYSMOD was also applied to other target zones created from the same distribution zone, these data sets for the other target zones are not cleaned up.

To delete the entries from these data sets, you can accept the SYSMOD and name these other target zones as the related zone. However, this updates the distribution library each time, which is time-consuming and can use up space in the distribution library data set. Instead, you can use the CLEANUP command, which deletes entries from the SMPMTS, SMPSTS, and SMPSCDS without updating the distribution library.

- ACCEPT followed by APPLY: When a SYSMOD is applied after it has been accepted, the entries associated with it are not deleted from the SMPMTS, SMPSTS, and SMPSCDS. To delete these entries, you can use the CLEANUP command.

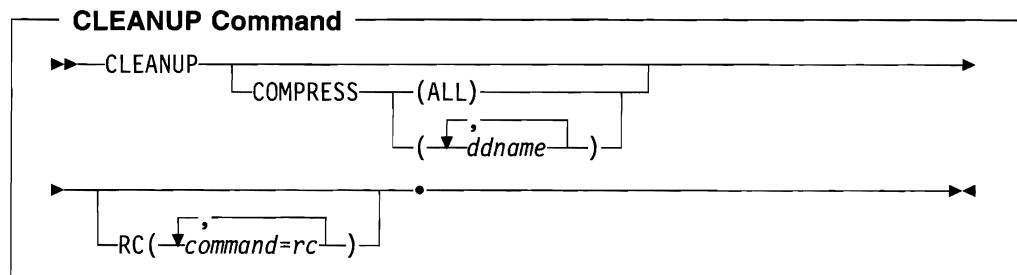
---

### Zones for SET BOUNDARY

For the CLEANUP command, the SET BOUNDARY command must specify the target zone associated with the SMPMTS, SMPSTS, or SMPSCDS data sets that should be cleaned up.

---

### Syntax



## Operands

### COMPRESS

indicates the ddnames of the data sets to be compressed after the entries are deleted. The valid data sets are SMPMTS, SMPSCDS, and SMPSTS.

- If you specify **ALL**, all three data sets are compressed.
- If you specify one or more specific ddnames, only the data sets they apply to are compressed.

**Note:** **COMPRESS** can also be specified as **C**.

### RC

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the CLEANUP command.

Before SMP/E processes the CLEANUP command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the CLEANUP command. Otherwise, the CLEANUP command fails. For more information about the RC operand, see Appendix D.

#### Notes:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the CLEANUP command. Therefore, you must specify every command whose return code you want SMP/E to check.

---

## Data Sets Used

The following data sets can be needed to run the CLEANUP command. They can be defined by DD statements or, ordinarily, by DDDEF entries. See Chapter 34 for more information about these data sets.

SMPCSI	SMPLOGA	SMPRPT	SMPSTS	SYSUT2
SMPCNTL	SMPMTS	SMPSCDS	SYSPRINT	SYSUT3
SMPLOG	SMPOUT	SMPSNAP	SYSUT1	<i>zone</i>

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are allocated dynamically by use of the ZONEINDEX information in the GLOBALZONE entry.

---

## Output

Two reports are produced during CLEANUP processing:

- CLEANUP Summary report
- File Allocation report

See Chapter 31 for descriptions of these reports.



### Example: Using CLEANUP with the COMPRESS Operand

Assume that you want to clean up the data sets for target zone MVSTGT and at the same time compress the SMPMTS data set. Before cleanup, the data sets contain the entries shown in Table 3.

*Table 3. CLEANUP Example: Data Set Members before CLEANUP Processing*

Data Set	Members
SMPSCDS	UZ00010 UZ00020
SMPMTS	MAC01 MAC02 MAC03 MAC04
SMPSTS	SRC11' SRC12 SRC13

Table 4 shows the information contained in the MVSTGT zone about the MAC and SRC entries.

*Table 4. CLEANUP Example: Service Levels of Elements*

Element	FMID	RMID	UMID
MAC01	JXY1234	UZ00010	AZ00100
MAC02	JXY1234	UZ00020	
MAC03	JXY1234	UZ00030	
MAC04	JXY2300	JXY2300	
SRC11	JXY1234	UZ00025	
SRC12	JXY1234	UZ00025	
SRC13	JXY2300	JXY2300	

Table 5 shows the information contained in the related distribution zone about the SYSMODs associated with those MAC and SRC entries.

*Table 5. CLEANUP Example: Status of SYSMODs*

SYSMOD	Accepted	Superseded By
AZ00100		
JXY1234	Yes	
JXY2300		
UZ00010	Yes	
UZ00020	Yes	
UZ00025		UZ00030
UZ00030	Yes	

You can use the following command to clean up and compress the data sets:

```
SET      BDY(MVSTGT)      /* Process MVSTGT tgt zone. */.  
CLEANUP COMPRESS(SMPMTS) /* Compress MTS at cleanup. */.
```

SMP/E deletes the entries shown in Table 6 from the data sets:

<i>Table 6. CLEANUP Example: Entries Deleted by CLEANUP Processing</i>	
<b>Data Set</b>	<b>Entries Deleted</b>
SMPSCDS	UZ00010 UZ00020
SMPMTS	MAC02 MAC03
SMPSTS	SRC11 SRC12

SMP/E then compresses the SMPMTS and writes a CLEANUP Summary report.

---

## Processing

The CLEANUP command deletes entries from the SMPSCDS, SMPMTS, and SMPSTS data sets.

SMP/E opens the specified target zone and its related distribution zone for read access. It also opens the SMPSCDS, SMPMTS, and SMPSTS for update processing.

- For the SMPSCDS, SMP/E checks which SYSMODs have BACKUP entries. It deletes the entries for each SYSMOD that has been accepted into the distribution library or that has been superseded by an accepted SYSMOD.
- For the SMPMTS, SMP/E checks which macros have MTSMAC entries in that data set and, for each entry, checks which SYSMODs are specified as the FMID, UMID, and RMID. It deletes the entries (and associated aliases) whose associated SYSMODs have been accepted into the distribution library or have been superseded by an accepted SYSMOD.
- For the SMPSTS, SMP/E checks which source elements have STSSRC entries in that data set and, for each entry, checks which SYSMODs are specified as the FMID, UMID, and RMID. It deletes the entries whose associated SYSMODs have been accepted into the distribution library or have been superseded by an accepted SYSMOD.

SMP/E then generates a report listing all the deleted entries and writes it to the SMPRPT data set. It compresses any data sets, if requested, and then closes them.

If there are no entries in any of the data sets, SMP/E does no cleanup, but does compress any data sets if requested. It then closes the data sets and writes a report to SMPRPT.

If SMP/E could not open a data set, or if the target zone has no record of a particular entry in one of the data sets, SMP/E issues an error message, and CLEANUP processing fails.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of CLEANUP processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

- Global zone – Read without enqueue.
- Target zone – Read without enqueue.
- DLIB zone – Read without enqueue.

### 2. Processing

- Target zone – Read without enqueue.
- DLIB zone – Read without enqueue.

### 3. Termination

All resources are freed.



---

## Chapter 5. The CONVERT Command

### Users of SMP/E Release 5 or later

You can skip this chapter. You do not need to convert any data sets to use them with SMP/E Release 8.1.

Because data elements are used in packaging SMP/E Release 8.1, if you are currently at SMP4, or SMP/E Release 4, or earlier, you cannot migrate directly to SMP/E Release 8.1. As soon as SMP/E Release 8.1 is available in an MVS CBIPO, however, you can order and install a CBIPO MVS feature that includes SMP/E Release 8.1. You can then use the SMP/E CONVERT command to convert any data sets from your previous release that will be used with SMP/E Release 8.1.

- If you are currently using SMP/E Release 4, or earlier, you must convert the SCDS and CSI data sets to SMP/E Release 8.1 format.
- If you are currently using SMP4, you must convert the following:
  - SMP4 SCDS data sets to SMP/E Release 8.1 format
  - SMP4 CDS, CRQ, ACDS, and ACRQ, data sets into CSI zones
  - SMP4 PTS to the global zone in a CSI data set and to an SMP/E PTS data set

A single CONVERT command can be used to convert:

- A single CSI or SCDS from a previous release of SMP/E
- A combination of SMP4 data sets

After converting these data sets, you can use SMP/E to install functions and service on your system.

### Notes:

1. Once you have converted a data set, the new format can be processed by SMP/E Release 8.1 but not by the release of SMP/E or SMP4 from which you migrated. However, the original data set can still be processed by that earlier release of SMP/E or SMP4.
2. You cannot use CONVERT for individual zones within a CSI (unless you have one zone per CSI). CONVERT processes **all** the zones within a CSI.

## Zones for SET BOUNDARY

To process the CONVERT command, the SET BOUNDARY command must specify the appropriate zone type:

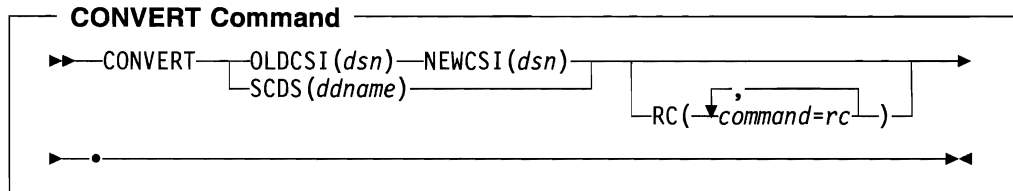
- For the PTS or CSI, the SMP/E Release 8.1 global zone must be specified.
- For the CDS, CRQ, or SCDS, the associated SMP/E Release 8.1 target zone must be specified.
- For the ACDS or ACRQ, the associated SMP/E Release 8.1 distribution zone must be specified.

## Syntax

This section shows the syntax for the following:

- Converting CSI and SCDS data sets from SMP/E Release 4 or an earlier release
- Converting SMP4 data sets

### Converting CSI and SCDS Data Sets (SMP/E Release 4 or Earlier)

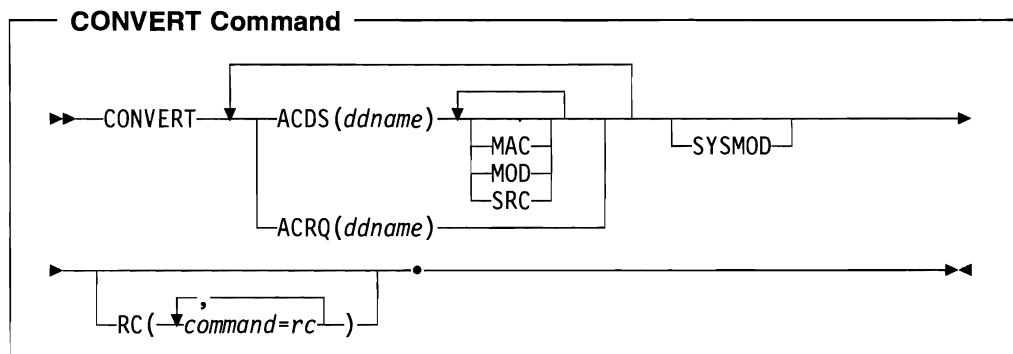


### Converting SMP4 Data Sets

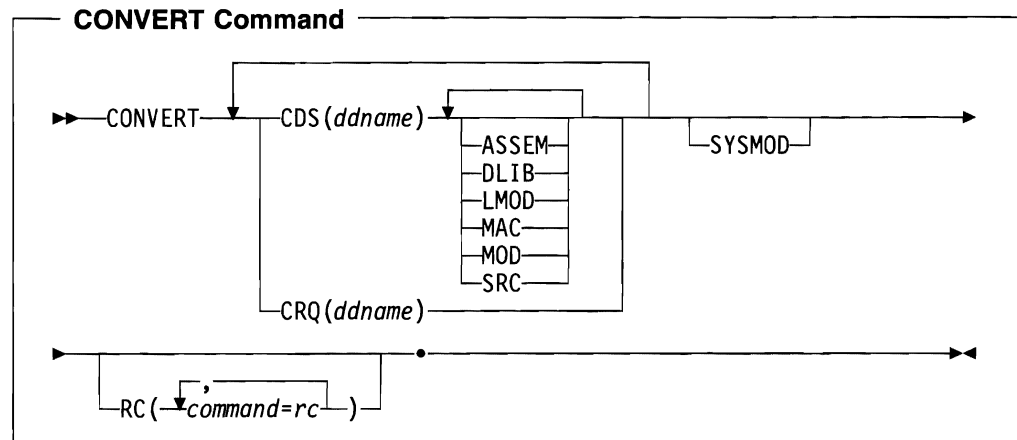
This section shows the syntax for converting:

- ACDS and ACRQ data sets
- CDS and CRQ data sets
- PTS data sets
- SCDS data sets

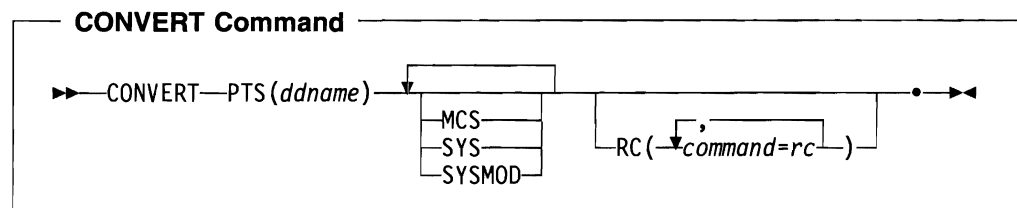
#### ACDS and ACRQ Data Sets



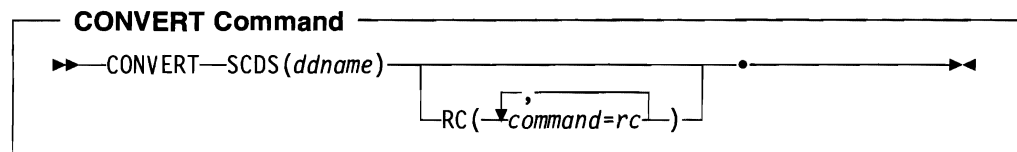
**CDS and CRQ Data Sets**



**PTS Data Sets**



**SCDS Data Sets**



**Operands**

**ACDS**

specifies the ddname of the ACDS data set that should be converted.

**Note:** ACDS is mutually exclusive with NEWCSI, OLDCSI, PTS, CRQ, CDS, and SCDS.

**ACRQ**

specifies the ddname of the ACRQ data set that should be converted.

**Note:** ACRQ is mutually exclusive with NEWCSI, OLDCSI, PTS, CRQ, CDS, and SCDS.

**ASSEM**

indicates that ASSEM entries should be converted.

**Note:** ASSEM is allowed only when you are converting the CDS.

**CDS**

specifies the ddname of the CDS data set that should be converted.

**Note:** CDS is mutually exclusive with NEWCSI, OLDCSI, PTS, ACRQ, and ACDS.

### CRQ

specifies the ddname of the CRQ data set that should be converted.

**Note:** CRQ is mutually exclusive with NEWCSI, OLDCSI, PTS, ACRQ, and ACDS.

### DLIB

indicates that DLIB entries should be converted.

**Note:** DLIB is allowed only when you are converting the CDS.

### LMOD

indicates that LMOD entries should be converted.

**Note:** LMOD is allowed only when you are converting the CDS.

### MAC

indicates that MAC entries should be converted.

**Note:** MAC is allowed only when you are converting the CDS or ACDS.

### MCS

indicates that MCS entries should be converted.

**Note:** MCS is allowed only when you are converting the PTS.

### MOD

indicates that MOD entries should be converted.

**Note:** MOD is allowed only when you are converting the CDS or ACDS.

### NEWCSI

specifies the data set name of the new CSI data set that will contain the converted data. The data set name must contain from 1 to 44 characters and cannot be split across input lines. This data set must be cataloged and primed with a GIMZPOOL record.

#### Notes:

1. If you specify **NEWCSI**, you must also specify **OLDCSI**.
2. **NEWCSI** and **OLDCSI** are mutually exclusive with **SCDS**, **ACDS**, **ACRQ**, **CDS**, **CRQ**, and **PTS**.
3. The data set must be empty except for the GIMZPOOL record.
4. If **OLDCSI** contains a global zone, **NEWCSI** must specify the same data set as the **SMPCSI DD** statement.
5. If **OLDCSI** does not contain a global zone, **NEWCSI** cannot specify the same data set as the **SMPCSI DD** statement.

### OLDCSI

specifies the data set name of the old CSI data set being converted. The data set name must be from 1 to 44 characters long, and cannot be split across input lines.



**Notes:**

1. If you specify **OLDCSI**, you must also specify **NEWCSI**.
2. **OLDCSI** and **NEWCSI** are mutually exclusive with **SCDS**, **ACDS**, **ACRQ**, **CDS**, **CRQ**, and **PTS**.
3. If the global zone is in a different CSI from the one that contains the target or **DLIB** zones, the CSI containing the global zone must be converted **before** the CSIs containing the other zones.

**PTS**

specifies the ddname of the PTS data set that should be converted. The ddname cannot be **SMPPTS**, which must be the ddname for the SMP/E PTS data set. Also, the data set name of the SMP/E PTS must be different from that of the PTS being converted.

**Notes:**

1. **PTS** is mutually exclusive with **NEWCSI**, **OLDCSI**, **SCDS**, **ACDS**, **ACRQ**, **CDS**, and **CRQ**.
2. When the SMP4 PTS is converted, internal **HOLDDATA** is not converted. PTFs with internal **HOLDDATA** that are applicable to SMP/E and that were received with SMP4 should be rejected and received with SMP/E. For more information, see the description of receiving **HOLDDATA** in the *SMP/E R8.1 Program Directory (English Feature)*.

**RC**

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the **CONVERT** command.

Before SMP/E processes the **CONVERT** command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the **RC** operand. If so, SMP/E can process the **CONVERT** command. Otherwise, the **CONVERT** command fails. See Appendix D for more information about the **RC** operand.

**Notes:**

1. The **RC** operand must be the **last** operand specified on the command.
2. If you do specify the **RC** operand, return codes for commands not specified do not affect processing for the **CONVERT** command. Therefore, if you use the **RC** operand, you must specify every command whose return code you want SMP/E to check.

**SCDS**

specifies the ddname of the **SCDS** data set that should be converted. The ddname **cannot** be **SMPSCDS**, which must be the ddname for the SMP/E Release 8.1 **SCDS** data set. Also, the new **SCDS** and the one being converted must have different data set names.

**Notes:**

1. SCDS is mutually exclusive with ACDS, ACRQ, NEWCSI, OLDCSI, and PTS.
2. Either a DD statement or DDDEF entries can be used for both the old SCDS to be converted and the SMP/E Release 8.1 SCDS to receive the converted data. If DDDEF entries are used, they must be in the target zone specified by the SET BOUNDARY command.
3. The new SCDS for SMP/E Release 8.1 must be empty.
4. If the old SCDS contains members created by SMP/E Release 5 or later, those members are copied to the new SCDS when the old SCDS is converted.

**SRC**

indicates that SRC entries should be converted.

**Note:** SRC is allowed only when you are converting the CDS or ACDS.

**SYS**

indicates that SYSTEM entries should be converted.

**Note:** SYS is allowed only when you are converting the PTS.

**SYSMOD**

indicates that SYSMOD entries should be converted.

**Note:** SYSMOD is allowed only when you are converting the ACDS, ACRQ, CDS, CRQ, or PTS.

## Syntax Notes

- You must specify at least one data set type (OLDCSI and NEWCSI; SCDS; or ACDS, ACRQ, CDS, CRQ, or PTS).
- If you are converting SMP4 data sets other than the SCDS, you can specify the entry types to be converted. Table 7 shows which entry types you can specify with which data set types.

*Table 7. Combinations of SMP4 Data Sets and Entry Types*

Entry Type	CDS	CRQ	ACDS	ACRQ	PTS
ASSEM	Yes				
DLIB	Yes				
LMOD	Yes				
MAC	Yes		Yes		
MCS					Yes
MOD	Yes		Yes		
SRC	Yes		Yes		
SYS					Yes
SYSMOD	Yes	Yes	Yes	Yes	Yes

**Note:** If you do not specify any entry type, SMP/E converts all the entries in the data set.

An entry type operand is applicable to all the data set operands specified on the CONVERT command. For example, SMP/E converts the SYSMOD entries in both the CDS and CRQ if you code the following:

```

SET      BDY(MVSTST1) /* Process MVSTST1 tgt zone. */.
CONVERT  CDS(OLDCDS) /* Convert CDS                */.
          CRQ(OLDCRQ) /* and CRQ                */.
          SYSMOD      /* SYSMOD entries.       */.
    
```

## Data Sets Used

The following data sets may be needed to run the CONVERT command. They can be defined by DD statements or, ordinarily, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPPTS	<i>zone</i>
SMP_CSI	SMPOUT	SMPSNAP	<i>ddname</i>
SMPLOG	SMPSCDS		

### Notes:

1. The SMPPTS data set is required only if the PTS operand is specified.
2. The SMPSCDS data set is required only if the SCDS operand is specified.
3. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are allocated dynamically, by use of the ZONEINDEX information in the GLOBALZONE entry.  
  
If a CSI other than the master CSI is being converted, *zone* represents the DD statement for the CSI that is being converted.
4. *ddname* is the name specified in the ACDS, ACRQ, CDS, CRQ, PTS, or SCDS operands.

## Usage Notes

The following sections contain additional information to help you use the CONVERT command:

- “Converting CSI Data Sets”
- “Converting SCDS Data Sets” on page 121
- “Reorganizing the CSI after Conversion” on page 122
- “Recovering After Conversion Fails” on page 123

## Converting CSI Data Sets

Before you convert any CSI data sets, do the following:

1. Determine the CSIs you need to convert. Remember:
  - You must convert each old CSI into its own new CSI.
  - You must convert all related data sets (CSI and SCDS data sets) before running any commands that use them. For example, if associated zones are in different CSIs, you must convert all those CSIs.

**Note:** The easiest way to keep all these data sets synchronized is to convert them in the same job. However, you are not required to do this. **Remember, a job that converts all these data sets at the same time takes a long time to run.**

- If you convert a CSI containing a global zone (a master CSI) and the ZONEINDEX subentries point to other CSIs, you must also convert those other CSIs. However, the master CSI must be converted first.
  - If you convert a CSI containing a target zone, but not the related distribution zone, you must also convert the CSI containing the distribution zone.
  - Likewise, if you convert a CSI containing a distribution zone, but not the related target zone, you must also convert the CSI containing the target zone.
  - You cannot convert more than one master CSI in the same job step.
2. If any zones in the CSIs are no longer required, use the ZONEDELETE command to delete them. (For example, you may no longer need zones for a test system that have been migrated to the production system.) This reduces the amount of data to be converted.
  3. Before you convert a CSI that contains a global zone, use the REJECT PURGE command to delete from the global zone and SMPPTS all SYSMODs that have been accepted into all the applicable related distribution zones. This reduces the number of SYSMODs to be converted.
  4. For each CSI that will be converted, allocate a new CSI for SMP/E Release 8.1 and initialize it with the SMP/E Release 8.1 GIMZPOOL record. Do **not** define any entries in the new CSI.

**Note:** Make sure to specify the correct key and record length for a Release 8.1 CSI. You **must** specify **KEYS(24 0)** and **RECORDSIZE(24 143)** when allocating the data set.

For more information about allocating CSI data sets, see the *SMP/E R8.1 User's Guide*.

If you plan to convert any SCDS data sets together with your CSI data sets, also follow the steps described under “Converting SCDS Data Sets” on page 121.

Now you are ready to convert your CSI data sets. See “Examples” on page 123.

**Notes:**

1. A single CONVERT command processes a single CSI. Therefore, you need a separate CONVERT command for each CSI you are converting.
2. Once you have started converting a system, do not change any of the old CSI data sets. For example, do not add, delete, or rename any zones.
3. If you have converted the CSI containing the global zone for a system but have not yet converted all the other related CSI data sets, do not run any SMP/E Release 8.1 commands that might try to access unconverted zones (for example, LIST ALLZONES).

## Converting SCDS Data Sets

Before you convert any SCDS data sets, do the following:

1. Convert the associated data sets to an SMP/E Release 8.1 target zone.
  - If you are converting from a previous release of SMP/E, convert the CSI containing the associated target zone.
  - If you are converting from SMP4, convert the CDS and CRQ data sets associated with the SCDS data sets.

**Note:** You can either convert the CSI or the CDS and CRQ data sets in separate jobs before you convert the SCDS, or convert the SCDS in the same job as these data sets. In the second case, SMP/E knows to convert the SCDS after it converts the related CSI or CDS and CRQ.

2. Delete from each SCDS all BACKUP entries no longer needed. This reduces the time required to convert the SCDS data sets. There are two ways to do this:
  - Accept all SYSMODs having entries in the SCDS that meet your criteria for being accepted.
  - Use the CLEANUP command to delete unnecessary entries from the SCDS. (The CLEANUP command is only available in SMP/E Release 3 and later.)

If the SCDS is now empty, there is no need to convert it. Instead, you can ignore the following steps, and use your SCDS with SMP/E Release 8.1.

3. For each SCDS to be converted, allocate a new SCDS for SMP/E Release 8.1.
4. Use DDDEF entries or DD statements to define the SCDS to be converted and the new Release 8.1 SCDS.
  - **For SMP/E conversions:** If the old target zone did not contain an SMPSCDS DDDEF entry, you must add an SMPSCDS DDDEF entry to define the SMP/E Release 8.1 SCDS, and another DDDEF entry to define the SCDS to be converted after you convert the CSI. You can use either the SMP/E Administration dialogs or the UCL ADD statement to add these entries.

If the old target zone already contained an SMPSCDS DDDEF entry, you must change that DDDEF entry to describe the SMP/E Release 8.1 SMPSCDS after you convert the CSI. You can use the SMP/E Administration dialogs or the UCL REP statement to change the data set name, and the volume and unit, if necessary.

To add a DDDEF entry for the converted SCDS, use the information from the original Release 4 SCDS DDDEF entry, but specify a different DDDEF name. You can use either the SMP/E Administration dialogs or the UCL ADD statement to add these entries.

- **For SMP4 conversions:** After you convert the CDS and CRQ to an SMP/E target zone, you must add an SMPSCDS DDDEF entry to define the SMP/E Release 8.1 SCDS, and another DDDEF entry to define the SCDS to be converted. These entries can be added using either the SMP/E Administration dialogs or the UCL ADD statement.

Now you are ready to convert your SCDS data sets. A later section of this chapter contains examples.

### Notes:

1. A single CONVERT command processes a single SCDS. Therefore, you need a separate CONVERT command for each SCDS you are converting.
2. Convert all CSI data sets related to the SCDS data sets before running any commands that use those data sets.

## Reorganizing the CSI after Conversion

During conversion of SMP4 data sets (other than the SCDS), VSAM inserts data from the data sets being converted into the appropriate area in the CSI data set. Sometimes VSAM requires a new control interval or control area to insert the data. When this occurs, it splits an existing interval or area and then inserts the new data. As a result of these control interval or control area splits, there may be some unused space in the CSI after conversion. You can compress this unused space by using the access method services with the EXPORT and IMPORT commands.

First, you allocate a sequential data set to contain the exported CSI. Then you run the EXPORT command to copy the CSI into the sequential data set. Remember to create a backup copy of the CSI before using the EXPORT command if you plan to use EXPORT to delete the old copy of the CSI after exporting it. Here is an example of the EXPORT command:

```
//EXPORT JOB 'accounting info',MSGLEVEL=(1,1)
//EXPORT EXEC PGM=IDCAMS
//TEMPFILE DD ....any sequential data set
//SMPCSI DD DSN=SMPE.SMPCSI.CSI,DISP=OLD
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
EXPORT SMPE.SMPCSI.CSI -
INFILE(SMPCSI) -
OUTFILE(TEMPFILE)
/*
```

Next, you allocate a new CSI data set to contain the imported CSI. Then you run the IMPORT command to copy the exported CSI into the new CSI data set. Here is an example of the IMPORT command:

```
//IMPORT JOB 'accounting info',MSGLEVEL=(1,1)
//IMPORT EXEC PGM=IDCAMS
//TEMPFILE DD ....same sequential data set used for export
//SMPCSI DD DSN=SMPE.SMPCSI.CSI,DISP=OLD
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
IMPORT -
INFILE(TEMPFILE) -
OUTFILE(SMPCSI) -
INTOEMPTY
/*
```

## Recovering After Conversion Fails

If there is a system failure during conversion of a CSI, there is no partial recovery of the data sets being converted. In that case, you must delete the SMP/E Release 8.1 CSI receiving the converted data, allocate and prime another SMP/E Release 8.1 CSI data set, and start the conversion process again with another CONVERT command.

This is also true for the SCDS data set. You must delete the SMP/E Release 8.1 SCDS that was receiving the converted data, allocate another SMP/E Release 8.1 SCDS data set, and start the conversion process again with another CONVERT command.

If a CONVERT command for other SMP4 data sets does not complete its processing because of a system or SMP/E failure, you can usually recover by rerunning the same job or a part of the same job. For more information, see “Example 9: Recovering After Conversion Fails” on page 136.

Although converting a large CDS or ACDS usually takes a long time, rerunning the same CONVERT command does not take the same amount of time. This is because when SMP/E stores data in a target zone or distribution zone, it first checks to see if the data is already there. If the data is there, SMP/E does not store the same data there a second time. This speeds up recovery by eliminating unnecessary I/O operations.

Another way to speed up recovery processing is to rerun the CONVERT command only for the entry types that were not successfully converted. To find the last entry that was successfully converted, look at the output from the failing CONVERT command and find the last message that says conversion was completed for an entry type.

**Note:** Although this method processes the second CONVERT command faster, it does require your time. In general, this effort is worthwhile only if you are converting the PTS. For example, if only the PTS MCS entries are successfully converted, you can improve CONVERT performance by rerunning the CONVERT command with only the SYSTEM and SYSMOD operands for the PTS.

---

## Output

The File Allocation report is produced during CONVERT processing. For a description of this report, see Chapter 31.

---

## Examples

The following examples are provided to help you use the CONVERT command:

- “Example 1: Converting a Single-CSI System” on page 124
- “Example 2: Converting a Multiple-CSI System” on page 127
- “Example 3: Converting a Master CSI Data Set” on page 130
- “Example 4: Converting a CSI Data Set for Target or DLIB Zones” on page 131

- “Example 5: Converting an SCDS Data Set” on page 131
- “Example 6: Converting the SMP4 PTS” on page 133
- “Example 7: Converting the SMP4 CDS and CRQ” on page 134
- “Example 8: Converting the SMP4 ACDS and ACRQ by Entry Type” on page 135
- “Example 9: Recovering After Conversion Fails” on page 136

Examples 6 through 8 show how to convert SMP4 data sets. They are all based on these assumptions:

- The target zone used for conversion is MVSTGT1 and the distribution zone is MVSDLB1. Both of these zones are defined in the GLOBALZONE ZONEINDEX and are described by TARGETZONE and DLIBZONE entries. For more information on defining these entries, see “GLOBALZONE Entry Syntax (Global Zone)” on page 364, “TARGETZONE Entry Syntax (Target Zone)” on page 374, and “DLIBZONE Entry Syntax (Distribution Zone)” on page 363.
- The data sets needed for SMP/E are defined by DDDEF entries or in a cataloged procedure named SMPPROC. For an example of a cataloged procedure, see Appendix B.
- The JCL in the examples is not complete. It shows only the DD statements for the SMP4 data sets.

### Example 1: Converting a Single-CSI System

Assume that you are converting from SMP/E Release 4 to SMP/E Release 8.1. You have a single CSI data set containing all the zones used to track the software on a given system, plus several related SCDS data sets.

All the Release 4 SCDS data sets are defined by DDDEF entries in your Release 4 target zones. As a result, each old SMPSCDS DDDEF entry is written to your new Release 8.1 target zones when you convert the CSI.

- You can use the UCL REP statement or the SMP/E Administration dialogs to change information in the copied DDDEF entries (such as the data set name, volume, and unit) to correspond to your new SCDS data sets for Release 8.1.

**Note:** The DDDEF name for each of the new SCDS data sets **must** be SMPSCDS.

- Assume that, for conversion, you want to add DDDEF entries for the Release 4 SCDS data sets to your Release 8.1 target zones. This way, both the old SCDS data sets and the new SCDS data sets are defined by DDDEF entries in your Release 8.1 target zones, instead of by DD statements. You can use the UCL ADD statement to define the DDDEF entry for your Release 4 SCDS.

**Note:** The DDDEF name for the old SCDS **cannot** be SMPSCDS.

- The CSI and the SCDS data sets should be converted in the same job to keep them synchronized. A separate CONVERT command is used for the CSI and for each SCDS.
- Because the OLDCSI data set contains the global zone, the NEWCSI operand **must** specify the same data set name as the SMPCSI DD statement.



Here is an example of a job that can be submitted to make the UCLIN changes and convert the CSI and SCDS data sets using the global zone, the IMS target zone, the NCP target zone, the CICS\* target zone, and the MVS target zone.

```
//CONVERT JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC specify program name or cataloged procedure
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SMPCNTL DD *
SET BDY(GLOBAL). /* Set to new global zone. */
CONVERT OLDCSI(REL4MVS.PROD.CSI)
NEWCSI(REL81MVS.PROD.CSI) /* Convert CSI. */
.
SET BDY(IMSTGT) /* Set to IMS target zone. */.
UCLIN /* */.
REP DDDEF(SMPSCDS) /* Define new SCDS. */
DA(REL81.IMS.SCDS) /* Make changes so that */
VOLUME(volume) /* this entry describes the */
UNIT(unit) /* SCDS for Release 8.1. */
OLD /* */.
ADD DDDEF(SCDS01) /* Define old SCDS. */
DA(REL4.IMS.SCDS) /* */
VOLUME(volume) /* */
UNIT(unit) /* */
OLD /* */.
ENDUCL /* */.
CONVERT SCDS(SCDS01) /* Convert old SCDS. */.
SET BDY(NCPTGT) /* Set to NCP target zone. */.
UCLIN /* */.
REP DDDEF(SMPSCDS) /* Define new SCDS. */
DA(REL81.NCP.SCDS) /* Make changes so that */
VOLUME(volume) /* this entry describes the */
UNIT(unit) /* SCDS for Release 8.1. */
OLD /* */.
ADD DDDEF(SCDS02) /* Define old SCDS. */
DA(REL4.NCP.SCDS) /* */
VOLUME(volume) /* */
UNIT(unit) /* */
OLD /* */.
ENDUCL /* */.
CONVERT SCDS(SCDS02) /* Convert old SCDS. */.
```

```

SET BDY(CICSTGT)          /* Set to CICS target zone. */.
UCLIN                    /*                               */.
REP      DDDEF(SMPSCDS)   /* Define new SCDS.          */
          DA(REL81.CICS.SCDS) /* Make changes so that    */
          VOLUME(volume) /* this entry describes the */
          UNIT(unit)      /* SCDS for Release 8.1.    */
          OLD             /*                               */.
ADD      DDDEF(SCDS03)    /* Define old SCDS.         */
          DA(REL4.CICS.SCDS) /*                               */
          VOLUME(volume) /*                               */
          UNIT(unit)      /*                               */
          OLD             /*                               */.
ENDUCL                    /*                               */.
CONVERT SCDS(SCDS03)     /* Convert old SCDS.        */.

SET BDY(MVSTGT)          /* Set to MVS target zone. */.
UCLIN                    /*                               */.
REP      DDDEF(SMPSCDS)   /* Define new SCDS.          */
          DA(REL81.MVS.SCDS) /* Make changes so that    */
          VOLUME(volume) /* this entry describes the */
          UNIT(unit)      /* SCDS for Release 8.1.    */
          OLD             /*                               */.
ADD      DDDEF(SCDS04)    /* Define old SCDS.         */
          DA(REL4.MVS.SCDS) /*                               */
          VOLUME(volume) /*                               */
          UNIT(unit)      /*                               */
          OLD             /*                               */.
ENDUCL                    /*                               */.
CONVERT SCDS(SCDS04)     /* Convert old SCDS.        */.
/*

```

**Notes:**

1. SMP/E updates the ZONEINDEX subentries in the GLOBALZONE entry to specify the new CSI data set names for the converted DLIB and target zones.
2. The input CSI and SCDS data sets remain unchanged.
3. If a Release 4 target zone did **not** contain an SMPSCDS DDDEF entry, no SMPSCDS DDDEF entry are be written to the associated Release 8.1 target zone when you convert the CSI. To define a DDDEF entry for the Release 8.1 SCDS, you can use either the UCL REP statement, as shown above, or the UCL ADD statement, specifying the same types of information as shown in the example for UCL REP.
4. You can use either DDDEF entries or DD statements for the old SCDS data sets to be converted. As with DDDEF entries, the ddname for each old SCDS **cannot** be SMPSCDS.

However, because this job needs more than one Release 8.1 SCDS data set (and the ddname of each new SCDS **must** be SMPSCDS), the Release 8.1 SCDS data sets must be defined by DDDEF entries in the appropriate target zones, rather than by DD statements.

Here is an example of a job you might run for the Release 4 CSI and the MVS SCDS data set if the Release 4 SCDS data sets were not defined by DDDEF entries in the old target zone. It specifies a DDDEF entry for the Release 8.1 MVS SCDS data set, and a DD statement instead of a DDDEF entry for the Release 4 MVS SCDS data set.

```
//CONVERT JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC specify program name or cataloged procedure
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SCDS04 DD DSN=REL4.MVS.SCDS,DISP=OLD
//SMPCNTL DD *
SET BDY(GLOBAL) /* Set to new global zone. */.
CONVERT OLDCSI(REL4MVS.PROD.CSI)
NEWCSI(REL81MVS.PROD.CSI) /* Convert CSI. */.

SET BDY(MVSTGT) /* Set to MVS target zone. */.
UCLIN /* */.
ADD DDDEF(SMPSCDS) /* Define new SCDS. */
DA(REL81.MVS.SCDS) /* */
VOLUME(volume) /* */
UNIT(unit) /* */
OLD /* */.
ENDUCL /* */.
CONVERT SCDS(SCDS04) /* Convert old SCDS. */.
/*
```

## Example 2: Converting a Multiple-CSI System

Assume that several CSIs are used to track the software on a given system. The master CSI (REL4MVS.PROD.CSI) contains the global zone and related MVS target and DLIB zones. There are separate CSIs containing target and DLIB zones for the IMS, NCP, and CICS subsystems. Related SCDS data sets also exist for each subsystem.

All the Release 4 SCDS data sets are defined by DDDEF entries in your Release 4 target zones. As a result, each old SMPSCDS DDDEF entry is written to the new Release 8.1 target zone when you convert each CSI.

- You can use the UCL REP statement or the SMP/E Administration dialog to change information in the copied DDDEF entries (such as the data set name, volume, and unit) to correspond to your new SCDS data sets for Release 8.1. The DDDEF name for each of the new SCDS data sets **must** be SMPSCDS.
- Assume that, for conversion, you want to add DDDEF entries for the Release 4 SCDS data sets to your Release 8.1 target zones. This way, both the old SCDS data sets and the new SCDS data sets are defined by DDDEF entries in your Release 8.1 target zones, rather than by DD statements. You can use the UCL ADD statement to define the DDDEF entry for your Release 4 SCDS. The DDDEF name for the old SCDS **cannot** be SMPSCDS.

You can convert all the CSI data sets plus the SCDS data sets in the same job to keep them synchronized. However, you may want to convert each CSI in a separate job, because a single job takes a long time to run. The examples below show a separate job for each CSI.

- On the CONVERT command for the global zone, the NEWCSI operand must specify the same data set name as the SMPCSI DD statement.

- On the CONVERT commands for the other zones, the NEWCSI operand must specify a different data set name from the one on the SMPCSI DD statement.

Here are examples of some jobs that can be submitted to make the UCLIN changes and convert a multiple-CSI system:

```
//CONVERT1 JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC specify program name or cataloged procedure
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SMPCNTL DD *
SET BDY(GLOBAL) /* Set to new global zone. */.
CONVERT OLDCSI(REL4MVS.PROD.CSI)
      NEWCSI(REL81MVS.PROD.CSI) /* Convert master CSI. */.
SET BDY(MVSTGT) /* Set to MVS target zone. */.
UCLIN /* */.
REP DDDEF(SMPSCDS) /* Define new SCDS. */
  DA(REL81.MVS.SCDS) /* Make changes so that */
  VOLUME(volume) /* this entry describes the */
  UNIT(unit) /* SCDS for Release 8.1. */
  OLD /* */.
ADD DDDEF(SCDSOLD) /* Define old SCDS. */
  DA(REL4.MVS.SCDS) /* */
  VOLUME(volume) /* */
  UNIT(unit) /* */
  OLD /* */.
ENDUCL /* */.
CONVERT SCDS(SCDSOLD) /* Convert old SCDS. */.
/*

//CONVERT2 JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC specify program name or cataloged procedure
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SMPCNTL DD *
SET BDY(GLOBAL) /* Set to new global zone. */.
CONVERT OLDCSI(REL4MVS.IMS.CSI)
      NEWCSI(REL81MVS.IMS.CSI) /* Convert IMS subsystem. */.
SET BDY(IMSTGT) /* Set to IMS target zone. */.
UCLIN . /* */.
REP DDDEF(SMPSCDS) /* Define new SCDS. */
  DA(REL81.IMS.SCDS) /* Make changes so that */
  VOLUME(volume) /* this entry describes the */
  UNIT(unit) /* SCDS for Release 8.1. */
  OLD /* */.
ADD DDDEF(SCDSOLD) /* Define old SCDS. */
  DA(REL4.IMS.SCDS) /* */
  VOLUME(volume) /* */
  UNIT(unit) /* */
  OLD /* */.
ENDUCL /* */.
CONVERT SCDS(SCDSOLD) /* Convert old SCDS. */.
/*
```

```

//CONVERT3 JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC specify program name or cataloged procedure
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SMPCNTL DD *
SET BDY(GLOBAL) /* Set to new global zone. */.
CONVERT OLDCSI(REL4MVS.NCP.CSI)
NEWCSI(REL81MVS.NCP.CSI) /* Convert NCP subsystem. */.
SET BDY(NCPTGT) /* Set to NCP target zone. */.
UCLIN /* */.
REP DDDEF(SMPSCDS) /* Define new SCDS. */
DA(REL81.NCP.SCDS) /* Make changes so that */
VOLUME(volume) /* this entry describes the */
UNIT(unit) /* SCDS for Release 8.1. */
OLD /* */.
ADD DDDEF(SCDSOLD) /* Define old SCDS. */
DA(REL4.NCP.SCDS) /* */
VOLUME(volume) /* */
UNIT(unit) /* */
OLD /* */.
ENDUCL /* */.
CONVERT SCDS(SCDSOLD) /* Convert old SCDS. */.
/*

//CONVERT4 JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC specify program name or cataloged procedure
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SMPCNTL DD *
SET BDY(GLOBAL) /* Set to new global zone. */.
CONVERT OLDCSI(REL4MVS.CICS.CSI)
NEWCSI(REL81MVS.CICS.CSI) /*Convert CICS subsystem.*/.
SET BDY(CICSTGT) /* Set to CICS target zone. */.
UCLIN /* */.
REP DDDEF(SMPSCDS) /* Define new SCDS. */
DA(REL81.CICS.SCDS) /* Make changes so that */
VOLUME(volume) /* this entry describes the */
UNIT(unit) /* SCDS for Release 8.1. */
OLD /* */.
ADD DDDEF(SCDSOLD) /* Define old SCDS. */
DA(REL4.CICS.SCDS) /* */
VOLUME(volume) /* */
UNIT(unit) /* */
OLD /* */.
ENDUCL /* */.
CONVERT SCDS(SCDSOLD) /* Convert old SCDS. */.
/*

```

**Notes:**

1. SMP/E updates the ZONEINDEX subentries in the GLOBALZONE entry to specify the new CSI data set names for the converted DLIB and target zones.
2. The input CSI and SCDS data sets are unchanged.
3. If a Release 4 target zone did **not** contain an SMPSCDS DDDEF entry, **no** SMPSCDS DDDEF entry is written to the associated Release 8.1 target zone when you convert the CSI. To define a DDDEF entry for the Release 8.1 SCDS, you can use either the UCL REP statement, as shown above, or the

UCL ADD statement, specifying the same types of information as in the example for UCL REP.

4. You can use either DDDEF entries, DD statements, or a combination of the two, for both the old SCDS to be converted and the Release 8.1 SCDS to receive the converted data. As with DDDEF entries, these rules apply:

- The ddname of the new SCDS **must** be SMPSCDS.
- The ddname for the old SCDS **cannot** be SMPSCDS.

Here is an example of a job you might run if a Release 4 SCDS was not defined by a DDDEF entry in the old target zone. It specifies a DDDEF entry for the Release 8.1 SCDS, and a DD statement instead of a DDDEF entry for the Release 4 SCDS.

```
//CONVERT1 JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC specify program name or cataloged procedure
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SCDSOLD DD DSN=REL4.MVS.SCDS,DISP=OLD
//SMPCNTL DD *
SET BDY(GLOBAL) /* Set to new global zone. */.
CONVERT OLDCSI(REL4MVS.PROD.CSI)
          NEWCSI(REL81MVS.PROD.CSI) /* Convert master CSI. */.
SET BDY(MVSTGT) /* Set to MVS target zone. */.
UCLIN /* */.
ADD DDDEF(SMPSCDS) /* Define new SCDS. */
    DA(REL81.MVS.SCDS) /* */
    VOLUME(volume) /* */
    UNIT(unit) /* */
    OLD /* */.
ENDUCL /* */.
CONVERT SCDS(SCDSOLD) /* Convert old SCDS. */.
/*
```

### Example 3: Converting a Master CSI Data Set

Assume that several CSI data sets are used to track the software on a given system, and the global zone is contained in REL4MVS.PROD.CSI (the master CSI). In this case, the data set containing the global zone is to be converted first, in a separate job. Data in SMP/E Release 6-format is written to the new CSI data set; the data set from the previous release will be unchanged. Because the CSI being converted contains a global zone, NEWCSI specifies the same data set as the SMPCSI DD statement. Here is an example of a job that can be submitted to do this:

```
//CONVERT JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC PGM=SMPPROC
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SMPCNTL DD *
SET BDY(GLOBAL) /* Set to new global zone. */.
CONVERT OLDCSI(REL4MVS.PROD.CSI)
          NEWCSI(REL81MVS.PROD.CSI) /* Convert master CSI. */.
/*
```

**Notes:**

1. If you decide to convert CSIs in separate jobs, you must convert the CSI containing the global zone **before** you convert any CSIs containing the associated DLIB and target zones.
2. The ZONEINDEX subentries from the old global zone are copied into the new global zone.
3. After you run the job, the ZONEINDEX subentries in the converted global zone still point to the unconverted CSIs for the target and DLIB zones. Those other CSIs must also be converted. Do not run any SMP/E Release 8.1 commands that access zones in those CSIs until those data sets have been converted.

**Example 4: Converting a CSI Data Set for Target or DLIB Zones**

Assume that you have just converted the master CSI containing the global zone, as in Example 3. Now you want to convert a secondary CSI data set (for example, the data set containing the target and DLIB zones for the IMS subsystem).

- There must already be a ZONEINDEX subentry for the secondary CSI in the converted global zone (pointed to by the SMPCSI DD statement).
- Because OLDCSI does not specify a CSI containing a global zone, NEWCSI specifies a different data set from the SMPCSI DD statement.

Here is an example of a job that can be submitted to convert the CSI:

```
//CONVERT JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC PGM=SMPPROC
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SMPCTL DD *
SET BDY(GLOBAL) /* Set to new global zone. */.
CONVERT OLDCSI(REL4MVS.IMS.PROD.CSI)
NEWCSI(REL81MVS.IMS.PROD.CSI) /* Convert secondary CSI. */.
/*
```

**Notes:**

1. If you decide to convert CSIs in separate jobs, you must convert the CSI containing the global zone **before** you convert any CSIs containing the associated DLIB and target zones.
2. When you run this job, the ZONEINDEX subentries in the master CSI (REL81MVS.PROD.CSI) are updated with the NEWCSI data set name for all target and DLIB zones being converted at this time.

**Example 5: Converting an SCDS Data Set**

Assume that you want to convert an SCDS in a separate job from its related CSI. The CSI containing the associated target zone must be converted **before** you convert the SCDS. (If you were converting from SMP4, the CDS and CRQ associated with the SCDS must be converted to an SMP/E target zone before you converted the SCDS.)

Assume that there was an SMPSCDS DDDEF entry in the Release 4 target zone. As a result, the old SMPSCDS DDDEF entry was written to your new Release 8.1 target zone when you converted the CSI.

- You can use the UCL REP statement or the SMP/E Administration dialog to change information in that copied DDDEF entry (such as the data set name, volume, and unit) to correspond to your new SCDS for Release 8.1.

**Note:** The DDDEF name for the new SCDS **must** be SMPSCDS.

- Also assume that, for conversion, you want to add a DDDEF entry for the Release 4 SCDS to your Release 8.1 target zone. This way, both the old SCDS and the new SCDS are defined by DDDEF entries in your Release 8.1 target zone, rather than by DD statements. You can use the UCL ADD statement to define the DDDEF entry for your Release 4 SCDS.

**Note:** The DDDEF name for the old SCDS **cannot** be SMPSCDS.

Here is an example of a job that can be submitted to make the UCLIN changes and convert the old SCDS:

```
//CONVERT JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC PGM=SMPPROC
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SMPCNTL DD *
SET BDY(TGT1) /* Set to new target zone. */.
UCLIN /* */.
REP DDDEF(SMPSCDS) /* Define new SCDS. */
DA(REL81.MVS.SCDS) /* Make changes so that */
VOLUME(volume) /* this entry describes the */
UNIT(unit) /* SCDS for Release 8.1. */
OLD /* */.
ADD DDDEF(SCDSOLD) /* Define old SCDS. */
DA(REL4.MVS.SCDS) /* */
VOLUME(volume) /* */
UNIT(unit) /* */
OLD /* */.
ENDUCL .
CONVERT SCDS(SCDSOLD) /* Convert old SCDS. */.
/*
```

### Notes:

1. If the Release 4 target zone did **not** contain an SMPSCDS DDDEF entry, **no** SMPSCDS DDDEF entry was written to your new Release 8.1 target zone when you converted the CSI. To define a DDDEF entry for the Release 8.1 SCDS, you can use either the UCL REP statement, as shown above, or the UCL ADD statement, specifying the same types of information as shown in the example for UCL REP.
2. You can use either DDDEF entries, DD statements, or a combination of the two, for both the old SCDS to be converted and the Release 8.1 SCDS to receive the converted data. As with DDDEF entries, these rules apply:
  - The ddname of the new SCDS **must** be SMPSCDS.
  - The ddname for the old SCDS **cannot** be SMPSCDS.



Here is an example of a job you might run if the Release 4 SCDS data set was not defined by a DDDEF entry in the old target zone. It specifies a DDDEF entry for the Release 8.1 SCDS, and a DD statement instead of a DDDEF entry for the Release 4 SCDS.

```
//CONVERT JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1 EXEC PGM=SMPPROC
//SMPCSI DD DSN=REL81MVS.PROD.CSI,DISP=OLD
//SCDSOLD DD DSN=REL4.MVS.SCDS,DISP=OLD
//SMPCNTL DD *
SET BDY(TGT1) /* Set to new target zone. */.
UCLIN /* */.
REP DDDEF(SMPSCDS) /* Define new SCDS. */
DA(REL81.MVS.SCDS) /* */.
VOLUME(volume) /* Make changes so that */.
UNIT(unit) /* this entry describes the */.
OLD /* SCDS for Release 8.1. */.
ENDUCL /* */.
CONVERT SCDS(SCDSOLD) /* Convert old SCDS. */.
/*
```

### Example 6: Converting the SMP4 PTS

The following CONVERT command can be used to convert the SMP4 PTS:

```
//JOB JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC SMPPROC
//OLDPTS DD DSN=SYS1.SMP4.SMPPTS,DISP=OLD
//SMPCNTL DD *
SET BDY(GLOBAL) /* Convert to global zone. */.
CONVERT PTS(OLDPTS) /* Convert PTS. */.
/*
```

To omit conversion of the PTS SYSTEM entry, you can use the following command:

```
//JOB JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC SMPPROC
//OLDPTS DD DSN=SYS1.SMP4.SMPPTS,DISP=OLD
//SMPCNTL DD *
SET BDY(GLOBAL) /* Convert to global zone. */.
CONVERT PTS(OLDPTS) /* Convert PTS */.
SYSMOD MCS /* SYSMODs and MCS. */.
/*
```

To reduce the amount of time required for an individual job, you can submit three separate jobs. However, total processing time is increased.

```
//JOB1    JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC    SMPPROC
//OLDPTS  DD DSN=SYS1.SMP4.SMPPTS,DISP=OLD
//SMPCNTL DD *
SET       BDY(GLOBAL)          /* Convert to global zone. */.
CONVERT  PTS(OLDPTS)          /* Convert PTS              */.
          SYS                  /* SYSTEM entry only.     */.

/*
//JOB2    JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC    SMPPROC
//OLDPTS  DD DSN=SYS1.SMP4.SMPPTS,DISP=OLD
//SMPCNTL DD *
SET       BDY(GLOBAL)          /* convert to global zone */.
CONVERT  PTS(OLDPTS)          /* convert PTS              */.
          SYSMOD              /* SYSMOD entries only    */.

/*
//JOB3    JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC    SMPPROC
//OLDPTS  DD DSN=SYS1.SMP4.SMPPTS,DISP=OLD
//SMPCNTL DD *
SET       BDY(GLOBAL)          /* convert to global zone */.
CONVERT  PTS(OLDPTS)          /* convert PTS              */.
          MCS                 /* MCS entries only        */.

/*
```

Each of these jobs (or set of jobs) has the same effect. The SMP4 PTS is converted to the CSI global zone and the SMP/E PTS.

### Example 7: Converting the SMP4 CDS and CRQ

The following CONVERT command can be used to convert both the CDS and the CRQ:

```
//JOB     JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC    SMPPROC
//OLDCDS  DD DSN=SYS1.SMP4.SMPCDS,DISP=OLD
//OLDCRQ  DD DSN=SYS1.SMP4.SMPCRQ,DISP=OLD
//SMPCNTL DD *
SET       BDY(MVSTST1)        /* Convert to new tgt zone. */.
CONVERT  CDS(OLDCDS)          /* Convert CDS              */.
          CRQ(OLDCRQ)         /* and CRQ.                 */.

/*
```

You can get the same result by using two CONVERT commands:

```
//JOB     JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC    SMPPROC
//OLDCDS  DD DSN=SYS1.SMP4.SMPCDS,DISP=OLD
//OLDCRQ  DD DSN=SYS1.SMP4.SMPCRQ,DISP=OLD
//SMPCNTL DD *
SET       BDY(MVSTST1)        /* Convert to new tgt zone. */.
CONVERT  CDS(OLDCDS)          /* Convert CDS.             */.
CONVERT  CRQ(OLDCRQ)         /* Convert CRQ.             */.

/*
```

To reduce the amount of time required for an individual job, you can submit two separate jobs. However, this increases total processing time.

```
//JOB1      JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC      SMPPROC
//OLDCDS    DD DSN=SYS1.SMP4.SMPCDS,DISP=OLD
//SMPCNTL   DD *
SET         BDY(MVSTST1)      /* Convert to new tgt zone. */.
CONVERT     CDS(OLDCDS)       /* Convert CDS.             */.
/*
//JOB2      JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC      SMPPROC
//OLDCRQ    DD DSN=SYS1.SMP4.SMPCRQ,DISP=OLD
//SMPCNTL   DD *
SET         BDY(MVSTST1)      /* convert to new tgt zone */.
CONVERT     CRQ(OLDCRQ).      /* convert CRQ              */.
/*
```

### Example 8: Converting the SMP4 ACDS and ACRQ by Entry Type

You can use the following CONVERT commands to convert the ACDS and ACRQ by entry types:

```
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC     SMPPROC
//OLDACDS  DD DSN=SYS1.SMP4.SMPACDS,DISP=OLD
//OLDACRQ  DD DSN=SYS1.SMP4.SMPACRQ,DISP=OLD
//SMPCNTL  DD *
SET        BDY(MVSDLB1)      /* Convert to new DLIB zone.*/.
CONVERT    ACDS(OLDACDS)     /* Convert ACDS             */.
MOD        /* MOD entries only. */.
CONVERT    ACDS(OLDACDS)     /* Convert ACDS             */.
MAC        /* MAC entries only. */.
CONVERT    ACDS(OLDACDS)     /* Convert ACDS             */.
SRC        /* SRC entries only. */.
CONVERT    ACDS(OLDACDS)     /* Convert ACDS             */.
SYSMOD     /* SYSMOD entries only. */.
CONVERT    ACRQ(OLDACRQ)     /* Convert ACRQ             */.
SYSMOD     /* SYSMOD entries only. */.
/*
```

As in the previous examples, you can submit five separate jobs to reduce the time required for an individual job. However, total processing time is increased.

## Example 9: Recovering After Conversion Fails

Assume that when SMP/E was converting the CDS, a system error occurred, causing SMP/E to abend. There are two ways to recover. The first is to rerun the same job. The second is to convert only the entries that were not successfully converted. For example, assume that SMP/E issued the following messages before abending:

```
GIM53401v CONVERT PROCESSING HAS STARTED FOR SYSTEM ENTRIES
           IN THE OLDCDS CDS LIBRARY
GIM53501v 00001 SYSTEM ENTRIES WERE CONVERTED FROM THE OLDCDS
           CDS LIBRARY
GIM53402v CONVERT PROCESSING IS COMPLETE FOR SYSTEM ENTRIES
           IN THE OLDCDS CDS LIBRARY
GIM53401v CONVERT PROCESSING HAS STARTED FOR ASSEM ENTRIES
           IN THE OLDCDS CDS LIBRARY
GIM53501v 00001 ASSEM ENTRIES WERE CONVERTED FROM THE OLDCDS
           CDS LIBRARY
GIM53402v CONVERT PROCESSING IS COMPLETE FOR ASSEM ENTRIES
           IN THE OLDCDS CDS LIBRARY
GIM53401v CONVERT PROCESSING HAS STARTED FOR LMOD ENTRIES
           IN THE OLDCDS CDS LIBRARY
GIM53501v 00001 LMOD ENTRIES WERE CONVERTED FROM THE OLDCDS
           CDS LIBRARY
GIM53402v CONVERT PROCESSING IS COMPLETE FOR LMOD ENTRIES
           IN THE OLDCDS CDS LIBRARY
GIM53401v CONVERT PROCESSING HAS STARTED FOR MACRO ENTRIES
           IN THE OLDCDS CDS LIBRARY
```

Because the CDS contains a SYSTEM entry and ASSEM, LMOD, MACRO, MODULE, SOURCE, DLIB, and SYSMOD entries, you can tell from these messages that LMOD entries were the last ones successfully converted. Therefore, you can rerun the CONVERT command, this time including operands for the remaining entry types. Here is an example of the second CONVERT job:

```
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//EXEC     SMPPROC
//OLDCDS   DD DSN=SYS1.SMP4.SMPCDS,DISP=OLD
//SMPCNTL  DD *
SET       BDY(MVSTST1)      /* Convert to new tgt zone. */.
CONVERT   CDS(OLDCDS)      /* Convert CDS                */.
           MAC              /* macro entries              */.
           MOD              /* MOD entries                 */.
           SRC              /* source entries              */.
           DLIB             /* DLIB entries                */.
           SYSMOD           /* SYSMOD entries.            */.
/*
```

**Note:** The MAC operand is required, because SMP/E did not issue a message saying that it completed conversion of the MAC entries.

## Processing

The type of processing depends on the type of data set being converted:

- CSI data sets (SMP/E only)
- SCDS data sets (SMP/E or SMP4)
- Other data sets (SMP4 only)

## CSI Processing

To convert a CSI data set, the SET command must specify the global zone in the SMP/E Release 8.1 CSI. (This data set is specified by the SMPCSI DD statement.) SMP/E makes sure that the operands of both OLDCSI and NEWCSI have been specified, and that no additional operands other than RC have been specified.

SMP/E also checks whether the CSI data sets are in the proper format. The old CSI data set must have been allocated with a KEYS value of (23 0) and a RECORDSIZE value of (23 142). The new CSI data set must have been allocated with a KEYS value of (24 0) and a RECORDSIZE value of (24 143), must have been initialized with a GIMZPOOL record, and must contain no other records or entries other than the one inserted by GIMZPOOL.

Next, SMP/E determines whether OLDCSI contains a global zone. If so, the value specified in NEWCSI must match the data set specified in the SMPCSI DD statement. Otherwise, NEWCSI must specify a data set other than the one specified by the SMPCSI DD statement.

If OLDCSI does not contain a global zone, SMP/E makes sure that the data set containing the global zone has already been converted. Specifically, SMP/E checks whether the global zone contains a ZONEINDEX subentry that specifies both of the following:

- The same zone name as the zone being converted
- The same data set name as the one specified on the OLDCSI operand

SMP/E then converts the data and writes it into the NEWCSI data set.

After conversion, SMP/E processes ZONEINDEX subentries.

- If the CSI being converted contained only a global zone, SMP/E copies the ZONEINDEX subentries from the old global zone to the new one.
- If the CSI being converted contained a target or DLIB zone, SMP/E updates the data set name in the associated ZONEINDEX subentry with the data set name specified by the NEWCSI operand.

## SCDS Processing

To convert an SCDS data set, the SET command must specify the SMP/E Release 8.1 target zone associated with the SCDS.

**Note:** The SCDS must be converted **after** the associated global and target zones have been converted. Otherwise, the SET command for the SCDS conversion fails.

SMP/E checks whether DD statements were provided for the old and new SCDS data sets. If no DD statement is found for an SCDS, SMP/E checks the specified

target zone for a DDDEF entry. The ddname or DDDEF name of the SCDS to be converted **cannot** be SMPSCDS. The ddname or DDDEF name of the Release 8.1 SCDS **must** be SMPSCDS.

SMP/E also checks whether the new SCDS is empty. If so, SMP/E converts members from the old SCDS that are not yet in SMP/E Release 8.1 format, and copies any members that are already in SMP/E Release 8.1 format.

## General Processing for Other Data Sets

Before SMP/E converts a data set other than a CSI or SCDS, it makes sure these conditions are met:

- At least one SMP4 data set must be specified.
- The specified data set must be compatible with the zone type specified in the previous SET command. Table 8 shows which data sets are converted into which zones.

*Table 8. SMP4 Data Sets and SMP/E Zone Types*

Zone Type	ACDS	ACRQ	CDS	CRQ	PTS
DLIB	Yes	Yes			
Global					Yes
Target			Yes	Yes	

- The zone specified in the SET command must already exist.
- If any of the entry type operands were specified, each one must be valid for at least one of the specified data sets. For a description of which entry types are valid for which SMP4 data sets, see Table 7 on page 118.
- There must be a DD statement for each ddname specified on the SMP4 data set type operands.

If any of these conditions is not met, SMP/E issues an error message, and no conversion is done.

If all the conditions are met, SMP/E converts the specified data sets. If several data sets are specified, they are converted in the following order:

CDS, then CRQ

or

ACDS, then ACRQ

**Note:** If you are converting one SMP4 data set at a time to improve performance, you should convert them in the above order.

SMP/E converts the specified entry types for each data set being converted. If no entry types have been specified, SMP/E converts each entry in each data set. Table 9 on page 139 shows the various types of SMP4 entries and the corresponding SMP/E entries.

**Note:** SMP/E adds a source ID of SMPECNV to each SYSMOD entry it has converted.

*Table 9. SMP4 Entries and the Corresponding SMP/E Entries*

SMP4 Data Set	SMP4 Entry	SMP/E Data Set	SMP/E Zone	SMP/E Entry
PTS	MCS	PTS	–	MCS
PTS	SYSMOD	CSI	GLOBAL	SYSMOD
PTS	SYSTEM	CSI	GLOBAL	GLOBALZONE UTILITY
CDS	ASSEM	CSI	TARGET	ASSEM
CDS	DLIB	CSI	TARGET	DLIB
CDS	LMOD	CSI	TARGET	LMOD
CDS	MAC	CSI	TARGET	MAC
CDS	MOD	CSI	TARGET	MOD
CDS	SRC	CSI	TARGET	SRC
CDS	SYSMOD	CSI	TARGET	SYSMOD
CDS	SYSTEM	–	–	Not converted
CRQ	FMID	–	–	Not converted
CRQ	SYSMOD	CSI	TARGET	SYSMOD
ACDS	MAC	CSI	DLIB	MAC
ACDS	MOD	CSI	DLIB	MOD
ACDS	SRC	CSI	DLIB	SRC
ACDS	SYSMOD	CSI	DLIB	SYSMOD
ACDS	SYSTEM	–	–	Not converted
ACRQ	FMID	–	–	Not converted
ACRQ	SYSMOD	CSI	DLIB	SYSMOD

SMP/E does not change the SMP4 data sets during conversion. These data sets are used only as input to the CONVERT command.

During conversion, many VSAM control interval splits occur. This creates a great deal of unused space in the CSI data set. After conversion, therefore, you should reorganize your CSI data set by exporting it, and then importing it. For an example of how to do this, see “Recovering After Conversion Fails” on page 123.

### Special Processing for Other Data Sets

Most of the entries in the SMP4 data sets are just stored in the new SMP/E data sets. However, the following entries are handled differently:

- The PTS SYSTEM entry
- The CDS/ACDS SYSTEM entry
- The CRQ/ACRQ FMID entry

**PTS SYSTEM Entry**

The information in the SMP4 PTS SYSTEM entry now resides in the SMP/E GLOBALZONE entry and in the various OPTIONS and UTILITY entries. The SREL list and the FMID list are stored in the GLOBALZONE entry. Information about the utilities SMP/E is to call is processed as follows:

- If the SYSTEM entry contained information for a utility, an SMP/E UTILITY entry is created and contains all the information for that utility. The name of the UTILITY entry matches the prefix used to describe that utility information in the SMP4 syntax. Table 10 shows the UTILITY entries that may be created during conversion.

*Table 10. UTILITY Entries Created by CONVERT*

<b>Utility Described in SYSTEM Entry for SMP4</b>	<b>Default Name for UTILITY Entry Created by SMP/E</b>
Assembler	ASM
Compress	COMP
Copy	COPY
Link-edit	LKED
Retry	RETRY
Update	UPDAT
Superzap	ZAP

- If a UTILITY entry already exists for a specified program name (or for the default name), that UTILITY entry is not changed. This way, the current information is not overlaid.

None of the other information in the PTS SYSTEM entry is processed during conversion. This is because that data is contained in OPTIONS entries, which must exist before you run SMP/E. In addition, SMP/E cannot determine which OPTIONS entries to store the data in.

**CDS or ACDS SYSTEM Entry**

The information in the SMP4 CDS or ACDS SYSTEM entry now resides either in an SMP/E TARGETZONE or DLIBZONE entry or in an OPTIONS entry. None of the information in the CDS or ACDS SYSTEM entry is converted, because the SMP/E entries must exist before you run SMP/E, and because SMP/E cannot determine which OPTIONS entries to store the data in.

**CRQ or ACRQ FMID Entry**

In SMP4 the FMID entries were used as an index into the SYSMOD entries during the installation of a new function. These entries indicated which requisite SYSMODs were needed to prevent the regression of service.

In SMP/E, this information is stored in the target zone and distribution zone SYSMOD entries. Because the CSI containing these entries is a VSAM data set, FMID entries are no longer needed as an index. The information in the CRQ and ACRQ SYSMOD entries (which are converted) is sufficient to create appropriate



target zone and distribution zone SYSMOD entries. Therefore, the CRQ and ACRQ FMID entries are not processed during conversion.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of CONVERT processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone – Read without enqueue.

### 2. Conversion

**Note:** This applies to the zones specified on the OLDCSI and NEWCSI operands, as well as to the zone specified on the SET command.

The zone or data set used depends on the data set being converted.

Global zone – Update with exclusive enqueue.

Target zone – Update with exclusive enqueue.

DLIB zone – Update with exclusive enqueue.

SMPPTS – Update with exclusive enqueue.

### 3. Termination

All resources are freed.



## Chapter 6. The DEBUG Command

With the DEBUG command, you can request diagnostic processing from SMP/E—for example:

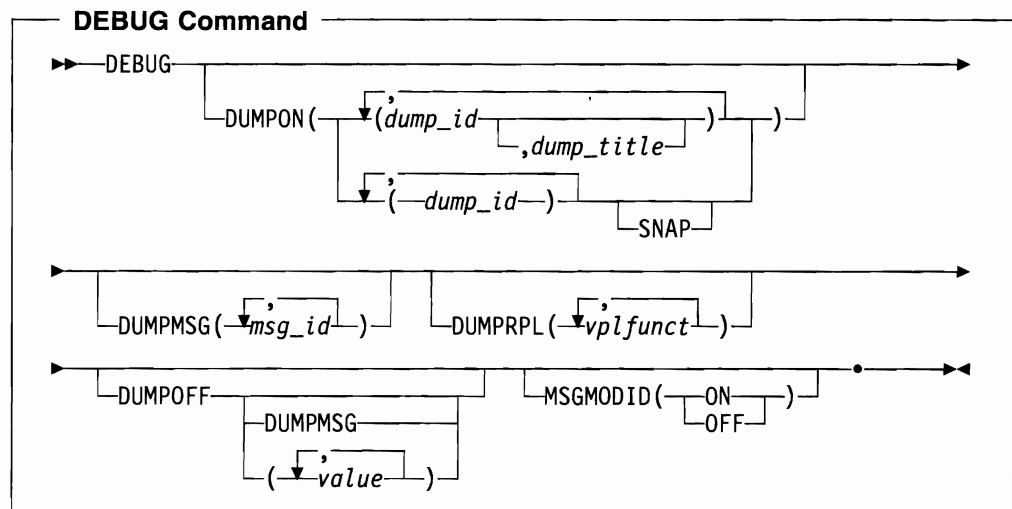
- Tracking the source of all SMP/E messages
- Dumping SMP/E control blocks and storage areas
- Dumping VSAM RPL control blocks
- Dumping SMP/E storage whenever specified messages are issued

You can use this command to provide additional documentation when reporting an SMP/E problem or when working with IBM to solve an SMP/E problem.

### Zones for SET BOUNDARY

The DEBUG command is used to collect diagnostic information for problems with other SMP/E commands. Therefore, you should use the same SET BOUNDARY command for DEBUG as for the other commands you are debugging.

### Syntax



### Operands

**Note:** The DEBUG dump operands are for use when working with IBM to solve an SMP/E problem, not before you report a problem. Therefore, some of the information you need to specify on those operands is provided through IBM and is not included in this book.

### DUMPON

specifies one or more dump points for which associated control blocks and storage areas are to be dumped. Unless **SNAP** is specified, the dump is formatted and written to the SMPDEBUG data set. These are the values you can specify on the DUMPON operand:

#### *dump-id*

is a predefined dump point, whose name is provided by IBM. You must specify at least one dump point.

#### *dump-title*

is an optional, user-written title for SMP/E to include on the header page of all formatted dumps requested by the DUMPON operand and written to the SMPDEBUG data set. The dump title may be up to 100 characters. If it contains parentheses, right and left parentheses must be in matched pairs. If **SNAP** is also specified, the dump title is not printed.

### SNAP

indicates that the dump is to be written unformatted to the SMPSNAP data set. If **SNAP** is specified, any dump title specified is not printed.

### DUMPOFF

specifies one or more dump points for which dumps are to be stopped. These are the values you can specify on the DUMPOFF operand:

#### **blank**

stops dumping for all dump points, including DUMPMSG.

#### **DUMPMSG**

stops dumping for all messages.

#### *value*

stops dumping for the specified dump point or VPLFUNCT value, which was provided by IBM. This dump point must have been activated by a previous DEBUG DUMPON or DEBUG DUMPRPL command.

**Note:** You can combine dump points and VPLFUNCT values on the same DEBUG DUMPOFF command.

### DUMPMSG

indicates that a SNAP dump is to be taken of SMP/E storage when the specified SMP/E messages are issued. *msg-id* is the message ID without the severity level, such as GIM44301. You must specify at least one message ID.

### DUMPRPL

indicates that a dump of the VSAM RPL control block and additional RPL information is to be taken. *vplfunct* is a VPLFUNCT value supplied by IBM. You must specify at least one VPLFUNCT value. The RPL dump is written to the SMPDEBUG data set. The heading for the RPL dump contains the VPL function being performed when the dump was taken. This can be used to separate the dumps if you specify more than one VPLFUNCT value on the DEBUG command.

**MSGMODID**

indicates whether to start or stop tracking which modules are issuing SMP/E messages.

**Note:** **MSGMODID** can also be specified as **M**.

**ON**

starts message tracking. Each SMP/E message is preceded by the name of the issuing module and the offset where the message was issued.

**OFF**

stops message tracking.

## Syntax Notes

- You must specify at least one operand.
- If you specify **DUMPON** or **DUMPRPL** and **DUMPOFF** on the same **DEBUG** command, these operands are processed in the order they occur. If you specify the same dump point on both operands, the last specification is used.
- **DEBUG** commands are generally used in pairs. The first one starts **DEBUG** processing and the second stops it. However, the second **DEBUG** command is not required if **SMP/E** is to do the same **DEBUG** processing for all the commands following the **DEBUG** command.

## Data Sets Used

The following data sets may be needed to run the **DEBUG** command. You can define them by **DD** statements or, usually, by **DDDEF** entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPDEBUG	SMPLOGA	SMPSNAP
SMP_CSI	SMPLOG	SMPOUT	

## Output

The File Allocation report is produced during **DEBUG** processing. For a description of this report, see Chapter 31.

## Examples

The following examples are provided to help you use the **DEBUG** command:

- “Example 1: Tracing SMP/E Messages” on page 146
- “Example 2: Dumping Control Blocks and Storage Areas” on page 146
- “Example 3: Dumping a VSAM RPL Control Block” on page 147
- “Example 4: Dumping SMP/E Storage When Messages Are Issued” on page 147

## Example 1: Tracing SMP/E Messages

Suppose that a problem occurred when you ran the following SMP/E commands:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
APPLY    S(USR0001)        /* Apply user modification. */.
```

Because the problem appeared to be in SMP/E and not in the USERMOD, you decided to report it to IBM. To help IBM determine the cause of the problem, you should also rerun the job with the message trace on. You can use the following commands:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
DEBUG    MSGMODID(ON)     /* Start message trace. */.
APPLY    S(USR0001)        /* Apply user modification. */.
DEBUG    MSGMODID(OFF)    /* Stop message trace. */.
```

When you run this job, SMP/E precedes all messages for the APPLY command with the name and offset of the issuing module. This stops when the second DEBUG command is processed.

**Note:** The second DEBUG command is not required if no further SMP/E commands are to be traced. It is used in this example only to show that the trace can be turned on and off.

## Example 2: Dumping Control Blocks and Storage Areas

Suppose that SYSMOD UZ12345 should have been selected when you ran the following SMP/E commands, but was not:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
APPLY    PTFS              /* Apply PTFs for HBT1201. */.
          FORFMID(HBT1201) /*                */.
          GROUP            /*                */.
```

After you report the problem to IBM, you may be asked to rerun the job with certain dump points enabled, for example, dump point 1. This information helps IBM determine the cause of the problem. You may also want to give the dump a title that describes the problem. You can use the following commands:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
DEBUG    DUMPON ((1,SYSMOD /* Specify debug dump point */.
          UZ12345 NOT      /* and dump title. */.
          SELECTED FOR    /*                */.
          APPLY))         /*                */.
APPLY    PTFS              /* Apply PTFs for HBT1201. */.
          FORFMID(HBT1201) /*                */.
          GROUP            /*                */.
DEBUG    DUMPOFF(1)       /* Stop debug dump. */.
```

When you run this job, SMP/E formats and dumps the control blocks and storage areas associated with dump point 1, then writes the dump to the SMPDEBUG data set.

### Example 3: Dumping a VSAM RPL Control Block

Suppose that when you ran the following SMP/E commands, SMP/E issued message GIM27901 on a VSAM error:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
APPLY    PTFS              /* Apply PTFs for HBT1201.   */.
          FORFMID(HBT1201) /*                               */.
          GROUP            /*                               */.
```

After you report the problem to IBM, you may be asked to rerun the job and request a dump of the VSAM RPL control block. You can use the following commands:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
DEBUG    DUMPRPL(VPLEXT)  /* Debug dump for VPLEXT.   */.
APPLY    PTFS              /* Apply PTFs for HBT1201.   */.
          FORFMID(HBT1201) /*                               */.
          GROUP            /*                               */.
DEBUG    DUMPOFF(VPLEXT)  /* Debug dump off.          */.
```

When you run this job, a dump of the VSAM RPL control block, plus additional RPL information, is written to the SMPDEBUG data set. The heading for the RPL dump contains the VPL function being performed when the dump is taken (in this case, VPLEXT).

### Example 4: Dumping SMP/E Storage When Messages Are Issued

Suppose that when you ran the following SMP/E commands, SMP/E issued message GIM38201, and you need more information about the problem:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
APPLY    PTFS              /* Apply PTFs for HBT1201.   */.
          FORFMID(HBT1201) /*                               */.
          GROUP            /*                               */.
```

After you report the problem to IBM, you may be asked to rerun the job and have SMP/E dump its storage whenever message GIM38201 is issued. You can use the following commands:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone. */.
DEBUG    DUMPMSG(GIM38201) /* Debug dump for GIM38201. */.
APPLY    PTFS              /* Apply PTFs for HBT1201.   */.
          FORFMID(HBT1201) /*                               */.
          GROUP            /*                               */.
DEBUG    DUMPOFF(DUMPMSG) /* Debug dump off.          */.
```

When you run this job, SMP/E dumps its storage and work areas to the SMP SNAP data set.

### Processing

When SMP/E encounters the DEBUG command, it first checks whether the SMPDEBUG and SMPSNAP data sets exist. These are opened when the dump is about to be written. It then scans the command for valid operands.

- If you specified **DUMPON**, SMP/E checks whether the dump points are valid. Unless **SNAP** is specified, the control blocks and storage areas are formatted and written to the SMPDEBUG data set. Otherwise, they are written unformatted to SMPSNAP.
- If you specified **DUMPRPL**, a dump of the VSAM RPL control block plus additional RPL information is written to the SMPDEBUG data set when the specified VPL function is performed.
- If you specified **DUMPMMSG**, a SNAP dump of SMP/E storage is written when the specified messages are issued.
- If you specified **DUMPOFF**, dumps are stopped for the specified dump points, or for all dump points if none are specified.
- If you specified **MSGMODID(ON)**, all messages are prefixed with the following string:

*@module+X'offset'*

where:

**module**

is the name of the SMP/E module (without the GIM prefix) that issued the message.

**offset**

is the hexadecimal offset into the module where the message was issued.

- If you specified **MSGMODID(OFF)**, messages are no longer prefixed with the module name and offset.

When SMP/E finishes processing the command following DEBUG, the SMPDEBUG and SMPSNAP data sets are closed.

DEBUG processing fails if a DUMPON dump point is incorrect, a required DD statement is missing, an incorrect VPLFUNCT value is specified with DUMPRPL, or a DUMPOFF dump point was not already active.



---

## Chapter 7. The GENERATE Command

With the GENERATE command, you can create a job stream that builds a set of target libraries from a set of distribution libraries. In that way, it is similar to system generation. However, the GENERATE command offers several advantages over system generation:

- GENERATE helps you reinstall products without SYSGEN support.

System generation creates jobs to install only products that are included by the system generation macros. Products without this SYSGEN support are not included. As a result, when SYSGEN is used to create or replace a system, a number of products have to be reinstalled outside the SYSGEN process.

GENERATE can create jobs to install **all** the products defined in a target zone, regardless of whether the products have SYSGEN support. As a result, when GENERATE is used to create or replace a system, no products have to be reinstalled outside the generation process.

- GENERATE creates job streams that are processed more efficiently.

The format of the system generation job stream depends on how the system generation macros are coded. For example, the number of products being installed and any changes in the system generation process may cause utilities to be called inefficiently.

The format of the GENERATE job stream is based on an analysis of the target zone. One job is created for each target library. This reduces the number of utility calls for each data set and improves SMP/E performance by allowing the various utilities to run concurrently.

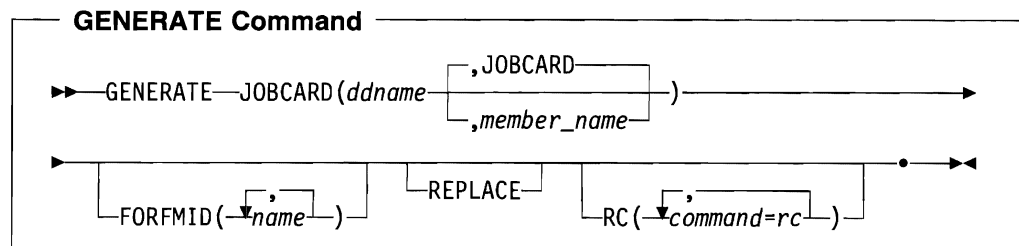
---

### Zones for SET BOUNDARY

For the GENERATE command, the SET BOUNDARY command must specify the target zone containing the entries used to create the job stream.

---

### Syntax



## Operands

### FORFMID

specifies the names of the FMIDs or FMIDSETs for which a job stream is to be generated. FORFMID should be used only if you want to create a job stream for specific products. To create a job stream describing all the products defined in the target zone, do not specify FORFMID.

**Note:** The FORFMID operand may include entries that do not have an FMID, such as the ASSEM entry. For more information, see “Determining Whether a Module Must Be Assembled” on page 157.

The products you can specify on the FORFMID operand depend on why you are using the GENERATE command:

- To initialize a new target zone from an old target zone by having the JCLIN command process the GENERATE output

In this case, you should specify only products without SYSGEN support for which JCLIN was not processed at ACCEPT time.

**Do not** specify any products for which you have done a stage 1 system generation. The stage 1 output should be processed by the JCLIN command to initialize the target zone.

**Note:** You do not need to specify any products for which you have processed inline JCLIN at ACCEPT time. Instead of using the GENERATE command to initialize the new target zone for these products, you should use the ZONECOPY or ZONEMERGE command to copy the distribution zone into the new target zone.

- To create installation job streams

In this case, you can specify any of the products defined in the target zone. This includes products with SYSGEN support, products without SYSGEN support, and products for which you have processed inline JCLIN at ACCEPT time.

### JOBCARD

indicates where SMP/E is to get the job card for the generated jobs. *ddname* is the ddname for the partitioned data set containing the job card member. *member-name* is the name of the member within that data set containing the job card. If *member-name* is not specified, the default is JOBCARD.

The JOBCARD operand is required.

### REPLACE

indicates that the JCL created should allow:

- Existing members in a data set to be replaced when the copy utility is invoked
- Existing load modules to be replaced when the link-edit utility is invoked

### RC

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the GENERATE command.

Before SMP/E processes the GENERATE command, it checks whether the return codes for the specified commands are less than or equal to the values

specified on the RC operand. If so, SMP/E can process the GENERATE command. Otherwise, the GENERATE command fails. For more information about the RC operand, see Appendix D.

**Notes:**

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the GENERATE command. Therefore, you must specify every command whose return code you want SMP/E to check.

## Data Sets Used

The following data sets may be needed to run the GENERATE command. They may be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPPUNCH	<i>zone</i>
SMP_CSI	SMPOBJ	SMPRPT	
SMPLOG	SMP_OUT	SMP_SNAP	

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are allocated dynamically by use of the ZONEINDEX information in the GLOBALZONE entry.

Besides checking the element entries in the CSI data set, SMP/E also uses information from the sources shown in Table 11 to create the output JCL for the GENERATE command. This information must, therefore, be defined before the GENERATE command is run.

<i>Table 11. Sources of Information for GENERATE Output JCL</i>	
Source of Input	JCL It Is Used for
JOB CARD member specified on the GENERATE command	JOB statement
UTILITY entries in the global zone (pointed to by the OPTIONS entry in effect for the target zone)	<ul style="list-style-type: none"> <li>• EXEC statements for utilities</li> <li>• ddnames for print output data sets</li> </ul>
DDDEF entries in the target zone	DD statements for: <ul style="list-style-type: none"> <li>• Distribution libraries (DLIBs)</li> <li>• Target libraries (SYSLIBs)</li> <li>• SMPWRK1—6 data sets</li> <li>• Print output data sets</li> <li>• SYSUT1—4 data sets</li> </ul>
Module GIMMPDFT	DD statements for: <ul style="list-style-type: none"> <li>• SMPWRK1—6 data sets</li> <li>• Print output data sets</li> <li>• SYSUT1—4 data sets</li> </ul>

### Usage Notes

Before using the GENERATE command, you must:

- Accept or restore all SYSMODs that have been successfully applied.
- Initialize the new target zone with the appropriate JCLIN.
- Define a target zone DDDEF entry for each distribution library and target library.
- Define the SMPPUNCH, SYSOUT, and temporary data sets required. You can use either DDDEF entries or update module GIMMPDFT.
- Define the SMPOUT and SMPRPT data sets. You either can use DDDEF entries or DD statements. If you have only an SMPOUT DD statement, the messages and reports are mixed together and are hard to read.

### Output

GENERATE output is written to the SMPPUNCH data set.

Two reports are produced during GENERATE processing:

- File Allocation report
- GENERATE Summary report

See Chapter 31 for descriptions of these reports.

### Examples

The following examples are provided to help you use the GENERATE command:

- “Example 1: Using GENERATE to Install New Products”
- “Example 2: Reinstalling Products Not Included by SYSGEN” on page 153

#### Example 1: Using GENERATE to Install New Products

Suppose that you want to install a new product, such as MVS/XA, into an existing set of distribution libraries, and then create a new set of target libraries to be controlled from target zone NEWMVS. You should follow these steps:

1. Install dummy function SYSMODs, if required. See the program directory for the product for additional information.
2. Receive and accept the product and any cumulative service. You can simplify the installation of SYSMODs without SYSGEN support if they contain inline JCLIN by processing that JCLIN when you accept the SYSMODs. To do this, the ACCJCLIN indicator must be set in the DLIBZONE entry before the SYSMODs are accepted. For more information about setting the ACCJCLIN indicator, see “Inline JCLIN” on page 36.
3. Allocate new target libraries.
4. Do a stage 1 generation to get the JCLIN for the SYSGEN-supported products. The JCL produced describes the structure of these products.

5. Run GENERATE against the old target zone, using the FORFMID operand to specify all products not included in the system generation. The JCL produced describes the structure of these products.

If inline JCLIN was processed for any of these products at ACCEPT time, do not specify them on the GENERATE command.

6. Allocate a new target zone.
7. Copy the distribution zone into the target zone.

If you processed inline JCLIN for SYSMODs without SYSGEN support at ACCEPT time, the target zone now describes the structure of those products.

8. Add any necessary entries to the new target zone and to the global zone—for example, DDDEF entries in the new target zone and OPTIONS entries in the global zone.
9. Run JCLIN against the new target zone, using the stage 1 generation output JCL from step 4 as input. This updates the target zone with information on the structure of all products included by SYSGEN.

**Note:** If there are intersections between SYSGEN-supported products and products without SYSGEN support, process step 9 after step 10.

10. Run JCLIN against the new target zone, using the GENERATE output as input. This updates the target zone with information on the structure of the specified products without SYSGEN support.
11. Create a JOBCARD member (in this example, MYJOB). When running the GENERATE command, include a DD statement pointing to the data set containing that member (in this example, JOB).
12. Run GENERATE against the new target zone without the FORFMID operand, as in this example:

```
SET      BDY(NEWMVS)          /* Process NEWMVS tgt zone.  */
GENERATE                                /* Generate JCL with          */
                                /* JOBCARD(JOB,MYJOB) /* JOBCARD from data set.  */
```

This creates jobs to install all the products defined in the target zone.

13. Submit the jobs created by the GENERATE command.

## Example 2: Reinstalling Products Not Included by SYSGEN

A target zone may contain products that are installed by a system generation procedure, as well as products that are not. When a system is created or replaced by use of a SYSGEN, products in the target libraries that were not included in the SYSGEN must be reinstalled. The GENERATE command can help you reinstall them.

**Note:** If any elements (such as macros or modules) are common to a product with SYSGEN support and a product without it, you must make sure the proper version of the element is used. To do this, you should examine all SYSGEN and GENERATE output and edit it as necessary. You may also need to process the steps in an order different from that described below.

Here is a procedure you can follow to avoid having to reinstall products when you do a SYSGEN:

1. Accept or restore all applied SYSMODs. This is to make sure that the distribution libraries used by SYSGEN are at the same service level as the current target zone.
2. Receive and accept the SYSMODs to be installed. You can simplify the installation of SYSMODs without SYSGEN support, if they contain inline JCLIN, by processing that JCLIN when you accept the SYSMODs. To do this, set the ACCJCLIN indicator in the DLIBZONE entry before accepting the SYSMODs.
3. Allocate new target libraries.
4. Do a stage 1 generation to get the JCLIN for the SYSGEN-supported products. The JCL produced describes the structure of these products.

5. Run GENERATE against the old target zone, using the FORFMID operand to specify all products that are not included in the system generation. The JCL produced describes the structure of these products.

If inline JCLIN was processed for any of these products at ACCEPT time, do not specify them on the GENERATE command.

6. Allocate a new target zone.
7. Copy the distribution zone into the new target zone. If you processed inline JCLIN for SYSMODs without SYSGEN support at ACCEPT time, the target zone now describes the structure of those products.
8. Add any necessary entries to the new target zone and to the global zone—for example, DDDEF entries in the new target zone and OPTIONS entries in the global zone.
9. Run JCLIN against the new target zone, using the stage 1 generation output JCL from step 4 as input. This updates the target zone with information on the structure of all products included by SYSGEN.

**Note:** If there are intersections between SYSGEN-supported products and products without SYSGEN support, process step 9 after step 10.

10. Run JCLIN against the new target zone, using the GENERATE output as input. This updates the target zone with information on the structure of the specified products without SYSGEN support.
11. Create a JOBCARD member (in this example, MYJOB). When running the GENERATE command, include a DD statement pointing to the data set containing that member (in this example, JOB).
12. Run GENERATE against the new target zone without the FORFMID operand, as in this example:

```
SET      BDY(NEWMVS)          /* Process NEWMVS tgt zone.  */.  
GENERATE                                /* Generate JCL with          */.  
        JOBCARD(JOB,MYJOB) /* JOBCARD from data set.    */.
```

This creates jobs to install all the products defined in the target zone.

13. Run the GENERATE jobs to build the system.

---

## Processing

The GENERATE command has three processing phases:

1. **Target zone analysis:** SMP/E analyzes the target zone to determine which modules, macros, source, load modules, data elements, and HFS elements must be created.
2. **JCL creation:** SMP/E constructs the JCL statements used to create the jobs to build the target libraries.
3. **Job generation:** SMP/E constructs the jobs necessary to create the target libraries.

## Target Zone Analysis

Before building the installation job stream, SMP/E first determines:

- Which elements are defined as part of the system
- Which target libraries the elements should be installed in
- Which utilities to use to install the elements

This information is in the target zone element, LMOD, ASSEM, and DLIB entries.

**Note:** SMP/E checks all the entries, even if **FORFMID** was specified on the GENERATE command. As a result, SMP/E may issue error messages for elements for which no install job is created. These can be ignored.

### Determining Which Macros to Install

To determine which macros to install and where to install them, SMP/E checks the MAC and DLIB entries. A macro should be installed if it meets all the following conditions:

- The macro has a DISTLIB subentry. DISTLIB indicates the distribution library containing the macro.
- The macro has a SYSLIB subentry, or the macro distribution library was totally copied during initial installation. If the library was copied, SMP/E looks for a SYSLIB subentry in the DLIB entry for the distribution library. SYSLIB indicates the target library where the macro is installed.

**Notes:**

1. Because of how SMP/E determines SYSLIB values, if the DLIB entry specifies more than one SYSLIB subentry, the value used for the macro might not be for the correct target library.
  2. Macros not residing in any SYSLIB are stored in the SMPMTS during APPLY and are deleted during ACCEPT. However, GENERATE does not create any steps to copy these macros to the SMPMTS, because when GENERATE is used, SYSMODs are not applied and then accepted. Rather, as in system generation, the SYSMODs are accepted, and then the elements are installed in the target libraries. As a result, the SMPMTS has no members.
- The FMID that owns the macro matches an FMID specified on the FORFMID operand. This is not checked if **FORFMID** was not specified.

**Note:** If **FORFMID** was not specified and SMP/E cannot determine which FMID owns the macro, the macro is still selected. If the macro is not found in the distribution library, however, an error may result.

SMP/E creates JCL to copy the macros from the distribution library into the target library.

### Determining Which Source to Install

To determine which source to install and where to install it, SMP/E checks the SRC and DLIB entries. A source should be installed if it meets all the following conditions:

- The source has a DISTLIB subentry. DISTLIB indicates the distribution library containing the source.
- The source has a SYSLIB subentry, or the source distribution library was totally copied during initial installation. If the library was copied, SMP/E checks for a SYSLIB subentry in the DLIB entry for the distribution library. SYSLIB indicates the target library where the source is installed.

#### Notes:

1. Because of how SMP/E determines SYSLIB values, if the DLIB entry specifies more than one SYSLIB subentry, the value used for the source might not be for the correct target library.
  2. Source not residing in any SYSLIB is stored in the SMPSTS during APPLY and deleted during ACCEPT. However, GENERATE does not create any steps to copy this source to the SMPSTS, because when GENERATE is used, SYSMODs are not applied and then accepted. Rather, as in system generation, the SYSMODs are accepted, and then the elements are installed in the target libraries. As a result, the SMPSTS has no members.
- The FMID that owns the source matches an FMID specified on the FORFMID operand. This is not checked if **FORFMID** was not specified.

**Note:** If **FORFMID** was not specified and SMP/E cannot determine which FMID owns the element, the source is still selected. If the source is not found in the distribution library, however, an error may result.

SMP/E creates JCL to copy the source from the distribution library into the target library.

**Note:** For more information about processing assembled modules, see the next section.

### Determining Which Modules to Install

To determine which modules to install and where to install them, SMP/E checks the MOD, LMOD, and DLIB entries. A module should be installed if it meets all the following conditions:

- The module has a DISTLIB subentry. DISTLIB indicates the distribution library or other data set containing the module.
- The module has at least one LMOD subentry, or the module distribution library was totally copied during initial installation. If the module contains an LMOD subentry, SMP/E checks for a SYSLIB entry in the corresponding LMOD entry. If the module is part of a totally copied library, SMP/E checks for a SYSLIB



subentry in the DLIB entry for the distribution library. SYSLIB indicates the target library where the module should be installed.

- The FMID that owns the module matches an FMID specified on the FORFMID operand. This is not checked if FORFMID was not specified.

**Notes:**

1. If the module does not have an FMID subentry, it may still be eligible if it is assembled by use of a macro owned by an FMID that was specified on the FORFMID operand. This is described in “Determining Whether a Module Must Be Assembled.”
2. If FORFMID was not specified and SMP/E cannot determine which FMID owns the element, the module is still selected. If the module is not found in the distribution library, however, an error may result. This is not an error if one product has linked a module from another product that resides in a different target zone.

Besides checking which modules to install and which libraries to install them in, SMP/E must also determine the following:

- Whether the module must be assembled before it is installed
- What load module to install the module into
- Whether to copy or link-edit the module

***Determining Whether a Module Must Be Assembled:*** To determine whether a module must be assembled, SMP/E checks whether the DISTLIB subentry is SYSPUNCH. If so, the module does not reside in a distribution library; it must then be assembled from some other source before it is link-edited. SMP/E must, therefore, determine how to assemble the module:

- If there is an ASSEM entry with the same name as the module, the statements in the ASSEM entry can be used to assemble the module.
- If there is no ASSEM entry, SMP/E checks for an SRC entry with the same name as the module. If one exists, the source can be used to assemble the module.
- If there are no ASSEM or SRC entries for the module, it cannot be assembled, nor can it be link-edited into any load modules.

If FORFMID was specified and the module being assembled does not have an FMID, it may still be eligible. If an ASSEM entry can be used to assemble the module, SMP/E checks whether that assembly uses macros owned by any of the specified FMIDs. If so, the module is included in the installation job stream. Otherwise, SMP/E assumes that the assembly is not needed for any of the selected FMIDs, and the module is not included in the installation job stream.

**Note:** To determine whether the ASSEM uses any selected macros, SMP/E checks all the MAC entries to see if any of them specify that ASSEM entry in the MAC GENASM list.

Because SYSPUNCH is used as a DISTLIB value for modules in the SMPOBJ data set, SMP/E also checks for a DD statement or DDDEF entry for SMPOBJ. This data set contains preassembled modules that can be used to avoid re-assembling the modules. If SMPOBJ is found, SMP/E checks whether it contains a member with the same name as the module that must be assembled. If so, the SMPOBJ

version of the module can be used with no need to reassemble the module. Otherwise, the module must be assembled.

**Determining Where and How a Module Should Be Installed:** To determine where and how the module should be installed, SMP/E checks the MOD entry, the LMOD entry, and the DLIB entry:

- If there is an LMOD subentry in the MOD entry, it indicates the load module that the module is part of. SMP/E checks the link-edit attributes in the corresponding LMOD entry to determine how to install the module.

If the attributes indicate that the module has been copied, SMP/E writes JCL to selectively copy the module into the target library. Otherwise, SMP/E uses the specified attributes and link-edit control statements in the entry to link-edit the load module into the target library.

- If there is no LMOD subentry in the MOD entry, the module is part of a totally copied library; therefore, there is no LMOD entry to check. In this case, SMP/E writes JCL to copy the entire distribution library into the target library. In addition, a SELECT statement is generated for each module.

When determining which load modules to link and how to link them, SMP/E checks the LMOD subentries in the MOD entries to ensure that each LMOD entry is referred to by at least one MOD entry. If any LMOD is found without a reference from a MOD, the load module is not selected for installation.

SMP/E also checks which modules are included in each load module. If a load module is to be link-edited, and not all of the modules in that load module were selected, SMP/E writes an INCLUDE card for the old version of the load module from the SYSLMOD data set. If all the modules for the load module are selected, the old version of the load module is not included. This enables the FORFMID operand to generate JCL that adds modules to a product that already exists in the target libraries.

### Notes:

1. If the module has cross-zone (XZLMOD) subentries, SMP/E does not try to create the associated cross-zone load modules. The GENERATE command creates JCL only for load modules in the set-to zone. You can use LINK to create these load modules.
2. If a load module contains cross-zone (XZMOD) subentries, SMP/E does not try to include the associated cross-zone modules. However, it does issue warning messages indicating which cross-zone modules were left out.

**Link-Editing When a Load Module Has a CALLLIBS Subentry List:** The link-edit steps for a load module with a CALLLIBS subentry list and its base version in the SMPLTS library are created as follows:

1. Building the base version of the load module in the SMPLTS library
  - INCLUDE statements are built for all modules in the distribution libraries that are explicitly included in the load module.
  - All the other link-edit control statements (except for ALIAS) that are defined for the load module are specified. To avoid a situation where a load module might have an alias with the same name as another load module, the ALIAS statements are **not** specified.

- All link-edit options defined in the LMOD entry are specified. NCAL is always passed.
2. Building the executable load module in the target libraries
    - INCLUDE statements are built for all modules in the distribution libraries that are explicitly included in the load module.
    - All other link-edit control statements in the LMOD entry are specified. CALL is always passed.
    - All link-edit options defined in the LMOD entry are specified.
    - A SYSLIB allocation, as defined by the CALLLIBS subentry list, is specified.

### Determining Which Data Elements to Install

To determine which data elements to install and where to install them, SMP/E checks the data element and DLIB entries. A data element should be installed if it meets all the following conditions:

- The data element entry has a DISTLIB subentry. DISTLIB indicates the distribution library containing the data element.
- The data element entry has a SYSLIB subentry, or the distribution library was totally copied during initial installation. In that case, SMP/E looks for a SYSLIB subentry in the DLIB entry for the distribution library. SYSLIB indicates the target library where the data element is installed.

**Note:** Because of how SMP/E determines SYSLIB values, if the DLIB entry specifies more than one SYSLIB subentry, the value used for the data element might not be for the correct target library.

- The FMID that owns the data element matches an FMID specified on the FORFMID operand. This is not checked if **FORFMID** was not specified.

**Note:** If **FORFMID** was not specified and SMP/E cannot determine which FMID owns the data element, the data element is still selected. If the data element is not found in the distribution library, however, an error may result.

SMP/E creates JCL to copy the data elements from the distribution library into the target library.

### Determining Which HFS Elements to Install

To determine which HFS elements to install and where to install them, SMP/E checks the HFS entries. An HFS element should be installed if it meets all the following conditions:

- The HFS entry has a DISTLIB subentry. DISTLIB indicates the distribution library containing the HFS element.
- The HFS entry has a SYSLIB subentry. SYSLIB indicates the “target library” (the hierarchical file system) in which the HFS element is installed.
- The FMID that owns the HFS element matches an FMID specified on the FORFMID operand. This is not checked if **FORFMID** was not specified.

**Note:** If **FORFMID** was not specified and SMP/E cannot determine which FMID owns the HFS element, the HFS element is still selected. If the HFS

element is not found in the distribution library, however, an error may result.

SMP/E creates JCL to copy the HFS elements from the distribution library into the target library.

### JCL Creation

After analyzing the target zone, SMP/E builds the following JCL statements for generating the jobs to install the elements:

- JOB card
- EXEC statement
- Distribution library and target library DD statements
- Utility work file DD statements
- Utility print output DD statements
- Other DD statements

#### JOB Card

The JOB card is obtained from the library and member specified on the JOBCARD operand of the GENERATE command. If the member name is not specified, "JOBCARD" is taken as the default member name. If the job card member is not found, message GIM64001 is issued.

The job card member can contain any number of records. However, the first record in the member must be the job card, and there must be room for an 8-character job name. This is because SMP/E does not check the format of the job card when generating jobs. The generated job name is stored in columns 3 through 10 of the first card read from the job card member.

The following example shows the expected format of the job card:

```
//xxxxxxx JOB 'accounting info',  
//           .....  
//           .....
```

#### EXEC Statement

The GENERATE command creates assembly, copy, and link-edit steps. The information used to construct the EXEC statement for each of these steps is obtained from the OPTIONS entry for the target zone. The OPTIONS entry points to a UTILITY entry for each utility that SMP/E can call. Each UTILITY entry contains:

- The name of the program to call.
- The parameters to pass that program. (Based on information in the target zone, SMP/E may add more parameters.)

Although you can specify up to 100 characters of data in the PARM field of a UTILITY entry, JCL restricts the length of the PARM field to a total of 100 characters. Therefore, only the first 50 characters of the UTILITY PARM field are used for the GENERATE command. The other 50 characters are reserved for SMP/E-generated parameters.

**Notes:**

1. The limitation on the length of the EXEC PARM field does not apply to invocations of the HFS copy utility. The parameters for this utility are specified on the EPARM operand of the generated SELECT statement instead of on the EXEC statement. However, the maximum total length of the parameters to be passed on the EPARM operand is X'FFFF' bytes. If the length exceeds this number, SMP/E truncates the parameters at the limit and passes this value to the HFS copy utility.
2. If the DFP level of the driving system on which SMP/E is running supports the binder, SMP/E supports PARM options exceeding 100 characters. In this case, SMP/E uses OPTIONS instead of PARM on the EXEC statement. GENOPTS is specified as the ddname, and a DD statement is created for GENOPTS. SMP/E then adds all other options after the GENOPTS DD statement. In this way, the PARM string limit of 100 characters can be exceeded for the binder link-edit step. SMP/E does not verify whether the DFP level of the system on which the GENERATE output is executed supports the binder.

If no OPTIONS entry is available, or if no UTILITY entries have been defined, the SMP/E default values are used. For a summary of these default values, see Table 30 on page 787.

**Distribution Library and Target Library DD Statements**

The information used to generate the DD statements for the target libraries and distribution libraries is obtained from the DDDEF entries in the target zone. Generally, the name of the DDDEF entry matches the name of the DD statement generated. Sometimes, however, SMP/E uses information in a DDDEF entry to generate a DD statement with a name different from the DDDEF entry name. For example, when calling the link-edit utility to link a load module to LINKLIB, SMP/E uses the LINKLIB DDDEF entry to generate the SYSLMOD DD statement.

**Utility Work File DD Statements**

The information used to generate the utility work files (SYSUT1 through SYSUT4) may be obtained from DDDEF entries or from entries in the temporary table in module GIMMPDFT. If there is no DDDEF entry for a work file, there must be an entry in GIMMPDFT. For more information about adding entries to GIMMPDFT, see "Module GIMMPDFT" on page 824.

When SMP/E generates assembly steps, it needs a temporary library to store the resulting object module. To generate the DD statement for this work file, SMP/E looks for a SYSPUNCH DDDEF entry or for a SYSPUNCH entry in module GIMMPDFT.

**Utility Print Output DD Statement**

The information used to generate the SYSOUT files (SYSPRINT) is obtained from DDDEF entries or from entries in the SYSOUT table in module GIMMPDFT. If there is no DDDEF entry for a SYSPRINT data set, and you do not want to use the SMP/E SYSOUT defaults, there must be an entry in GIMMPDFT. For more information about adding entries to GIMMPDFT, see "Module GIMMPDFT" on page 824.

### All Other DD Statements

SMP/E directly generates other DD statements for the input files (such as SYSIN and SYSLIN) for the various utilities.

## Job Generation

The final phase of GENERATE processing is the actual building of the installation jobs. These jobs are written to the SMPPUNCH data set. Jobs are created for the following:

- Elements to be copied
- Target libraries for link-edited load modules
- Load modules installed in libraries specified in a SYSLIB allocation
- Elements to be installed in a hierarchical file system
- Totally copied libraries

### Elements to Be Copied

One job, named COPYJOB, is produced for all copied elements. There is a separate COPY statement for each unique combination of distribution library, target library, and element type. The COPY and SELECT statements generated in this step are arranged by output library ddname, and by distribution library module name under each output library ddname.

#### Notes:

1. If you specified REPLACE on the GENERATE command, each COPY statement indicates that replacement is allowed.
2. GENERATE determines whether to list the names of copied members based on the value of the LIST subentry in the copy UTILITY entry in effect for the set-to target zone. For more information about the LIST subentry, see "UTILITY Entry (Global Zone)" on page 787.

### Target Libraries for Link-Edited Load Modules

One job is produced for each target library into which load modules must be link-edited. The name of the job matches the ddname of the target library; for example, the job to link-edit modules into SYS1.LINKLIB is LINKLIB. These jobs can contain multiple steps.

- The first steps are assemblies for any modules that are link-edited into this target library. These steps are named ASSM0001 through ASSM9999.
- Following the assembly steps are one or more link-edit steps for the target library. One link-edit step is generated for each unique set of link-edit attributes. These steps are named LINK0001 through LINK9999.

**Note:** If REPLACE was specified on the GENERATE command, the NAME statement for each load module indicates that replacement is allowed.

During any job step generation, if SMP/E cannot generate a required JCL statement, it issues message GIM64601 to identify the JCL statement that could not be generated, as well as the current job and step name. A single JCL comment is generated in place of the statement. The comment has the following format:

```
//*XXXXXXXX *** ERROR *** UNABLE TO GENERATE JCL STATEMENT
      FOR yyyyyyyy
```

where:

**XXXXXXXX**

is the name that would have been generated on the JCL statement.

**yyyyyyyy**

is the name of the DDDEF entry, SYSOUT table entry, or temporary table entry that should have been used to generate this JCL statement.

If SMP/E determines that no elements should be installed, it issues message GIM64901 or GIM64902.

### Load Modules Installed in Libraries Specified in a SYSLIB Allocation

When a load module specifies a SYSLIB allocation, two jobs are generated:

- An SMPLTS job is generated to link-edit the base version of the load module into the SMPLTS library before the creation of the final link-edit job. This job can also include load modules that are installed in target libraries that are part of another load module's SYSLIB allocation. SMP/E inserts a JCL comment statement (/\*SMPLTS) immediately after the SMPLTS job card to ensure that when the JCLIN command processes the GENERATE output, SMP/E can detect the SMPLTS job and skip over it.
 

**Note:** When SMP/E detects the /\*SMPLTS comment during JCLIN processing, it skips all the steps in the SMPLTS job. For this reason, the /\*SMPLTS comment should never be modified.
- LKSYSLIB is the final link-edit job. It is generated to link-edit the executable version of the load module into the target libraries. SMP/E inserts a JCL comment statement (/\*CALLLIBS=YES) immediately after the LKSYSLIB job to ensure that when the JCLIN command processes the GENERATE output, the SYSLIB DD statements in the output are processed and not ignored.

If there are no link-edits for load modules specifying a SYSLIB allocation, SMP/E does **not** create the SMPLTS and LKSYSLIB jobs.

### Elements to Be Installed in a Hierarchical File System

When elements need to be installed in a hierarchical file system (HFS), the HFSINST job is generated to invoke the HFS copy utility for those elements:

- One job step is generated per target library within the HFS.
- The name of each job step matches the ddname for the associated target library.
- Each job step installs multiple HFS elements from multiple distribution libraries into a single target library. This is done by invoking the SMP/E program GIMGNIAP, which calls the HFS copy utility to do the actual installation. For

more information about the control statements passed to GIMGNIAP, see Appendix J, "GIMGNIAP: HFS Copy Utility Invocation Program" on page 869.

- The DD statement for the target library specifies the pathname, not a data set name.
- COPY and SELECT statements are arranged by distribution library ddname and by element name under each distribution library ddname.
- The first two bytes after EPARM( on the SELECT control statement may appear as blanks or odd characters. However, they are valid data and should not be changed or deleted.

If no PATH value was specified in the DDDEF entry for a target library DD statement in the HFSINST job, SMP/E issues message GIM64602. Instead of generating the required JCL statement, SMP/E produces a comment with the following format:

```
//*XXXXXXXX *** ERROR *** INFORMATION MISSING FOR THIS DDNAME
```

where:

**XXXXXXXX**

is the name of the DDDEF entry that should contain a PATH subentry for the HFS element's target library.

### Totally Copied Libraries

The final job produced is a copy step for all totally copied distribution libraries. It contains one IEBCOPY COPY statement for each DLIB entry in the target zone. This step is not meant to be executed, because all the elements are selectively copied. It is, therefore, preceded by two // statements, which cause the reader or interpreter to ignore it. This step is generated so SMP/E can reconstruct the DLIB entries if the GENERATE output is processed by the JCLIN command.

## Using the Output from GENERATE

This section describes the following considerations for using the output job streams from the GENERATE command:

- Multiprogramming considerations for running the jobs concurrently
- Considerations for using the jobs to reinstall products

### Multiprogramming Considerations

The GENERATE command produces several jobs. The first, COPYJOB, does all the copies. You should run COPYJOB first so the subsequent jobs can run concurrently. This not only constructs all the macro and source libraries the other steps use, but prevents the subsequent jobs from using the data sets when COPYJOB needs them.

**Note:** If an LKSYSLIB job is generated, the link-edit jobs created by GENERATE might not be able to run concurrently. This can occur when the SYSLMOD data set specified in one of the steps in the LKSYSLIB job is the same as a SYSLMOD data set specified in another link-edit job.



## Reinstalling Products Using GENERATE

A target zone can contain products that are installed by a system generation procedure, as well as products that are not. Typically, when you run a generation procedure, such as SYSGEN, the products included by the generation macro are installed in a new set of target libraries. Any products not included by the generation macro must be reinstalled in those new libraries to make the system complete.

With the GENERATE command, you can avoid having to separately reinstall products not included in a generation procedure. To do this, first make sure the target zone for the new target libraries is updated with JCLIN that describes all the products that will be installed in those libraries.

- The JCLIN for the products included by the generation procedure is in the stage 1 generation output.
- The JCLIN for the products not included can be obtained by running the GENERATE command against the target zone for the current target libraries, and by specifying the desired products on the FORFMID operand.

If you processed inline JCLIN at ACCEPT time for any of these products, you can update the target zone with that JCLIN by copying the distribution zone into the target zone.

Once the new target zone describes all these products, you can run the GENERATE command without the FORFMID operand to create jobs that install all the products defined in that zone. You can then run these jobs to create the system, without having to separately reinstall any products. This procedure is described in more detail in “Example 2: Reinstalling Products Not Included by SYSGEN” on page 153.

---

## Zone and Data Set Sharing Considerations

The following are the phases of GENERATE processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

1. Initialization
 

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
2. Processing
 

Target zone	—	Read with shared enqueue.
-------------	---	---------------------------
3. Termination
 

All resources are freed.



---

## Chapter 8. The JCLIN Command

JCLIN processing initializes entries in the target or distribution zone with the data required to install elements into the target libraries. SMP/E derives this data by scanning a job stream containing assembly, copy, and link-edit job steps. JCLIN processing is called by either the JCLIN command or as part of installing a SYSMOD containing ++JCLIN statements.

SMP/E does **not** execute the JCLIN—this is important to note. It does not call the utilities specified in the job stream, and it does not update, modify, or look at any of the target libraries. Rather, JCLIN is a vehicle to describe the structure of the target libraries (and, ultimately, the structure of the load modules in those libraries). It is element statements (such as ++MOD) that cause SMP/E to actually invoke the utilities.

The purpose of this chapter is to describe JCLIN processing and the requirements SMP/E has on the input, so that you can better understand how to construct JCLIN for your own use.

### Notes:

1. JCLIN processes only information about elements such as macros, modules, and source. It does not process information about data elements, except for totally copied libraries. Information about data elements is added to the zone when the data elements are installed in the libraries or when the distribution zone is copied into the target zone.
2. JCLIN processing does **not** cause SMP/E to update the target or distribution libraries; only the entries in the target and distribution zones are updated. These libraries are updated when SMP/E processes the elements in a SYSMOD. The element statements in a SYSMOD determine which elements should be installed.

The following terms are used in this chapter:

**JCLIN input** A job stream of assembly, link-edit, and copy job steps. This input is pointed to by the SMPJCLIN DD statement.

**Inline JCLIN** JCLIN data pointed to by a ++JCLIN MCS. The actual JCLIN data can be packaged inline, following the ++JCLIN MCS, or it can be packaged in a RELFILE or TXLIB data set.

The description of the JCLIN command used here applies to processing both the JCLIN command and to inline JCLIN processed at APPLY or ACCEPT time. The utility and OPCODE operands that can be specified on the JCLIN command are also valid as operands on the ++JCLIN statement.

### Reminder

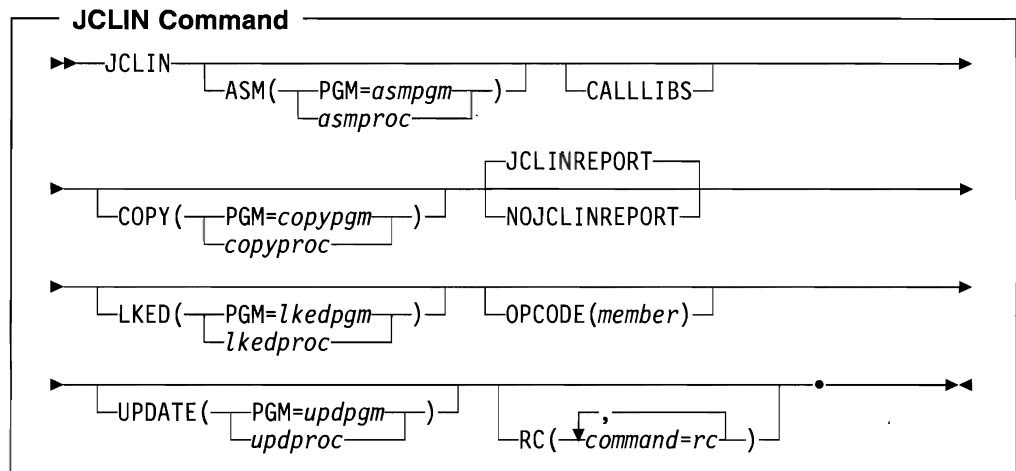
The input for JCLIN must be free of JCL errors and must conform to the SMP/E restrictions. Incorrect input can cause unpredictable errors, or even cause the installation of SYSMODs that depend on the JCLIN to fail.

## Zone for SET BOUNDARY

The zone specified on the SET BOUNDARY command depends on the type of JCLIN being processed.

- For the JCLIN command, the SET BOUNDARY command must specify the name of the target zone to be updated.
- For inline JCLIN processed by the APPLY command, the SET BOUNDARY command must specify the name of the target zone to be updated.
- For inline JCLIN processed by the ACCEPT command, the SET BOUNDARY command must specify the name of the distribution zone to be updated, and the ACCJCLIN indicator must be set in the DLIBZONE entry. For more information, see “Inline JCLIN” on page 36.

## Syntax



## Operands

### ASM

specifies an assembler program, *asmpgm*, or procedure, *asmproc*, in addition to those recognized by SMP/E. SMP/E recognizes programs ASMBLR, ASMA90, IEUASM, IEV90, and IFOX00, as well as procedure ASMS.

### CALLLIBS

specifies that SMP/E is to process SYSLIB DD statements in JCLIN link-edit steps. If the CALLLIBS operand is not specified on either the JCLIN command or the ++JCLIN statement, SMP/E JCLIN processing ignores any SYSLIB DD statements it encounters.

### COPY

specifies a copy program, *copypgm*, or procedure, *copyproc*, in addition to those recognized by SMP/E. SMP/E recognizes only the IEBCOPY program.

### JCLINREPORT

specifies that SMP/E is to write the JCLIN reports. This is the default.

**Note:** JCLINREPORT can also be specified as JCLR.

**LKED**

specifies a link-edit utility program, *lkedpgm*, or procedure, *lkedproc*, in addition to those recognized by SMP/E. SMP/E recognizes programs IEWL, HEWL, and IEWBLINK, as well as procedure LINKS.

**NOJCLINREPORT**

specifies that SMP/E is not to write any JCLIN reports.

**Note:** **NOJCLINREPORT** can also be specified as **NOJCLR**.

**OPCODE**

identifies a PARMLIB member containing control cards identifying character strings to be treated as operation codes or macros.

If you specify an OPCODE value, SMP/E uses the definitions in the associated PARMLIB member with those in GIMOPCDE, instead of using its default method of identifying macros and operation codes. (GIMOPCDE is the PARMLIB member provided by SMP/E.) Therefore, **all** character strings that must be identified as macros or operation codes must be defined either in GIMOPCDE or the specified PARMLIB member.

If no OPCODE value is specified, SMP/E uses GIMOPCDE to determine whether a character string is a macro or an operation code.

If SMP/E cannot find a PARMLIB member that defines macros and operation codes, it uses a default method for identifying character strings.

For more information about PARMLIB members and SMP/E's default method of identifying macros and operation codes, see "Building MAC Entries" on page 184 and Chapter 33.

**PGM**

specifies a program name (instead of a procedure name) for a utility.

**RC**

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the JCLIN command.

Before SMP/E processes the JCLIN command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the JCLIN command. Otherwise, the JCLIN command fails. For more information about the RC operand, see Appendix D.

**Notes:**

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the JCLIN command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

**UPDATE**

specifies an update program, *updpgm*, or procedure, *updproc*, in addition to those recognized by SMP/E. SMP/E recognizes only program IEBUPDTE.

**Note:** Although JCLIN does not actually derive any target zone initialization data from update job steps, update job steps are recognized so that the

update step SYSIN data (which may itself contain JCL) is not processed.

---

### Data Sets Used

The following data sets may be needed to run the JCLIN command. You can define them by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOG	SMP_OUT	SMP_SNAP
SMP_CSI	SMPLOGA	PARMLIB	<i>zone</i>
SMP_JCLIN			

#### Notes:

1. The PARMLIB DD statement is required only if the JCLIN input contains assembly steps.
2. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are allocated dynamically by use of the ZONEINDEX information in the GLOBALZONE entry.

---

### Usage Notes

This section describes the following:

- “SYSMODs with Inline JCLIN”
- “PARMLIB Members to Identify Opcodes in Assembler Text” on page 171
- “Packaging JCLIN Input” on page 171
- “Processing after System Generation or I/O Generation” on page 172
- “Cross-Zone Relationships” on page 172

### SYSMODs with Inline JCLIN

A SYSMOD can be constructed with the JCLIN necessary for it to be installed. In this case, the SYSMOD contains a ++JCLIN statement. When processing such a SYSMOD, SMP/E processes the JCLIN as part of the installation of the SYSMOD. During APPLY processing, and before changing any target zone entry affected by the JCLIN, a copy of that entry is stored on the SMPSCDS data set. This data is used by SMP/E in case the SYSMOD has to be restored. The target zone can then be brought back to the level it was at before the apply was run.

When applying multiple SYSMODs, each containing JCLIN, the JCLIN from any superseded SYSMODs is not processed.

**Note:** Each target zone must have its own SMPSCDS data set, because the information in the SMPSCDS data set is unique to that target zone. This SMPSCDS data set must also be used with the related distribution zone.

## PARMLIB Members to Identify Opcodes in Assembler Text

Because SMP/E uses information stored in a PARMLIB member (GIMOPCODE or a user-defined member) to differentiate macro invocations from assembler operation codes, the PARMLIB DD statement is required when SMP/E processes JCLIN with assembly steps.

Licensed programs or user products that use special assemblers recognizing additional operation codes must provide a PARMLIB member identifying those codes, and the name of that PARMLIB member must be specified in the OPCODE operand. For more information about defining PARMLIB members for JCLIN processing, see Chapter 33.

## Packaging JCLIN Input

Once JCLIN has been processed in the target zone by the SMP/E JCLIN command, it cannot be removed. Therefore, if there is any chance that you will want to remove the JCLIN changes, you should not use the JCLIN command in processing the input. Instead, you should package the JCLIN input as part of a SYSMOD, and then receive and apply the SYSMOD. The following is an example of such a SYSMOD:

```

++USERMOD(USR0001).
++VER(Z038) FMID(USRFUNC).
++JCLIN.
//...
//...
//... JCLIN input
//...
//...

```

These are the commands to install it:

```

SET      BDY(GLOBAL)      /* Set to global zone.      */.
RECEIVE  SYSMODS          /* Receive the SYSMOD.      */.
         S(USR0001)       /*                          */.
SET      BDY(MVSTST1)     /* Process MVSTST1 tgt zone.*/.
APPLY    S(USR0001)       /* Apply it.                 */.

```

When you use this method, any target zone entries that are changed as a result of the new JCLIN are first saved on the SMPSCDS data set. The SYSMOD, USR0001, can then be restored if you want to remove the JCLIN.

**Note:** After doing a full system generation, you must:

- Delete your old target zone and define a new one.
- Use ZONECOPY to copy the distribution zone to the new target zone.
- Scratch and reallocate your SMPMTS and SMPSTS data sets.

If you do not start your new system with an empty SMPMTS and SMPSTS, you may inadvertently regress your new system when service is applied.

## Processing after System Generation or I/O Generation

If you do an I/O generation, refer to the applicable system generation manual for further directions on required SMP/E processing.

After a complete system generation, you must copy the distribution zone, using an SMP/E zone utility command (such as ZONECOPY, ZONEEXPORT and ZONEIMPORT, or ZONEMERGE), to the new target zone before doing any JCLIN processing. This makes sure the initial target zone entries match those of the distribution zone and contain entries not created by JCLIN processing.

After a partial system generation (that is, a device generation), the output of stage 1 must be used as input to JCLIN processing to make sure that:

- Module, macro, and load module entries in the target zone are updated.
- New assembler entries are stored with the new assembler input in the target zone.
- Link-edit utility control statements for load module entries are automatically replaced, except for link-edit utility CHANGE and REPLACE control statements.

CHANGE and REPLACE control statements are associated with the DLIB module name found on the next INCLUDE statement in the link-edit job.

- If the same INCLUDE statement is contained in the stage 1 output, the CHANGE and REPLACE control statements are replaced.
- If that INCLUDE statement is **not** contained in the stage 1 output, the CHANGE and REPLACE control statements are carried over to the updated entry.

## Cross-Zone Relationships

If SMP/E creates an LMOD entry during JCLIN processing and there is already a copy of that entry containing only cross-zone subentries (a *stub* entry), SMP/E issues messages indicating that the cross-zone relationship defined by cross-zone subentries might no longer be valid. SMP/E then deletes the cross-zone subentries from the LMOD entry. If there is no longer a TIEDTO relationship between the cross-zone and the set-to zone, SMP/E also deletes the TIEDTO value for the cross-zone from the zone definition entry for the set-to zone. It does not, however, update related cross-zone modules to indicate that they are no longer part of the load module in the set-to zone or the cross-zone's TIEDTO subentry. You need to determine whether the cross-zone relationship is still valid. If it is valid, use the LINK command to reestablish it. For more information, see Chapter 9.

---

## Output

The following reports are produced during JCLIN processing:

- File Allocation report
- JCLIN Cross-Reference report
- JCLIN Summary report

For descriptions of these reports, see Chapter 31.



## Examples

The following examples are provided to help you use the JCLIN command:

- “Example 1: JCLIN for Products with Special Utilities”
- “Example 2: JCLIN for Products with Special Assembler Opcodes”
- “Example 3: JCLIN for MOD Entries” on page 174
- “Example 4: JCLIN for MAC and SRC Entries” on page 177
- “Example 5: JCLIN for an Assembler Step to Create a SRC Entry” on page 178
- “Example 6: JCLIN for Load Modules Residing in a Hierarchical File System” on page 178

### Example 1: JCLIN for Products with Special Utilities

Given the following JCLIN input:

```
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1    EXEC PGM=MYCOPY
//XYZDLIB DD DSN=SYS1.XYZDLIB,DISP=SHR
//XYZLOAD DD DSN=SYS1.XYZLOAD,DISP=SHR
//SYSIN    DD *
COPY INDD=XYZDLIB,OUTDD=XYZLOAD
/*
```

SMP/E processes it but does not recognize MYCOPY as a valid program name; therefore, it creates no entries.

But given the following JCLIN command:

```
SET      BDY(XYZTST1)      /* Process zone XYZTGT1.    */.
JCLIN    COPY(PGM=MYCOPY) /* Process JCLIN.      */.
```

SMP/E now recognizes the step as a copy step and builds a DLIB entry for XYZDLIB.

### Example 2: JCLIN for Products with Special Assembler Opcodes

Assume you have a special version of the assembler supporting two additional operation codes, LOOP and ENDLOOP, and that the following JCLIN is to be processed:

```
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//STEP     EXEC PGM=MYASM
//SYSPUNCH DD DSN=&&PUNCH(NEWMOD),
//          SPACE=(TRK,(1,1,1)),DISP=(,PASS)
//SYSIN    DD *
NEWMOD     CSECT
           .
           .
LOOPSTRT   LOOP
           .
           .
LOOPEND    ENDLOOP
           .
           .
           END NEWMOD
```

To process this correctly and have SMP/E recognize that LOOP and ENDLOOP are opcodes, you should set up the following PARMLIB member (in this case, named SMPPRM01):

```
KEY=LOOP    TYPE=OPCODE.
KEY=ENDLOOP TYPE=OPCODE.
```

The command to execute the JCLIN should then be as follows:

```
SET      BDY(XYZTST1)      /* Process zone XYZTST1.  */.
JCLIN    ASM(MYASM)        /* Special assembler.    */.
          OPCODE(SMPPRM01) /* Special opcode list.  */.
```

When SMP/E processes this input, it recognizes that MYASM is an assembler program, and that LOOP and ENDLOOP are operation codes.

### Example 3: JCLIN for MOD Entries

Given the job steps shown in this example, SMP/E creates target zone entries as follows:

```
//C1      EXEC PGM=IEBCOPY
//AMACLIB DD DSN=SYS1.AMACLIB,DISP=SHR
//MACLIB  DD DSN=SYS1.MACLIB,DISP=SHR
//SYSIN DD *
  COPY INDD=AMACLIB,OUTDD=MACLIB  TYPE=MAC
/*
//C2      EXEC PGM=IEBCOPY
//AOS14   DD DSN=SYS1.AOS14,DISP=SHR
//LINKLIB DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSIN DD *
  COPY INDD=AOS14,OUTDD=LINKLIB  TYPE=MOD
  SELECT MEMBER=((LOADMODC,,R))
/*
//DEFINE  JOB 'accounting info',MSGLEVEL=(1,1)
//A1      EXEC PGM=IEUASM
//SYSLIB  DD DSN=SYS1.AMACLIB,DISP=SHR
//SYSPUNCH DD DSN=&&PUNCH(TESTMOD1),
//          SPACE=(TRK,(1,1,1)),DISP=(,PASS)
//SYSIN DD *
TESTMOD1 CSECT
          TESTMAC1 --- INVOKE MACRO
          END TESTMOD1
/*
//L1      EXEC PGM=IEWL,PARM='LET,LIST,NCAL,RENT'
//SYSLMOD DD DSN=SYS1.LPALIB,DISP=SHR
//SYSPUNCH DD *.A1.SYSPUNCH,DISP=(SHR,PASS)
//SYSLIN  DD *
          INCLUDE SYSPUNCH(TESTMOD1)
          NAME LOADMOD1(R)
/*
```

```

//L2      EXEC PGM=IEWL,PARM='LET,LIST,NCAL'
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12   DD DSN=SYS1.AOS12,DISP=(SHR,PASS)
//SYSLIN DD *
          INCLUDE AOS12(TESTMOD2)
          INCLUDE AOS12(TESTMOD3)
          NAME LOADMODX(R)
/*
//L3      EXEC PGM=IEWBLINK,PARM='OL,AMODE=31,OPTIONS(OPTNAME)'
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12   DD DSN=SYS1.AOS12,DISP=SHR
//OPTNAME DD *
          FETCHOPT(PACK,PRIME),RMODE=31
          MAXBLK(256)
/*
//SYSLIN DD *
          INCLUDE AOS12(GIMMPDRV,GIMMPDR1)
          ENTRY GIMMPDRV
          SETCODE AC(1)
          NAME GIMMPP(R)
/*
//L4      EXEC PGM=IEWBLINK,PARM='CALL,RENT,REUS'
//SYSLMOD DD DSN=SYS1.APPLOAD,DISP=SHR
//AOS12   DD DSN=SYS1.AOS12,DISP=SHR
//SYSLIB  DD DSN=SYS1.V2R2M0.PLIBASE,DISP=SHR
//        DD DSN=SYS1.V2R2M0.APPBASE,DISP=SHR
//SYSLIN DD *
          INCLUDE AOS12(MOD00004,MOD00005)
          ENTRY MOD00004
          SETCODE AC(1)
          NAME LMOD04(R)
/*

```

### Copy Step C1

This copy step informs SMP/E that an entire distribution library has been copied to an operating system library. From the INDD operand, SMP/E determines the ddname of the distribution library and creates a target zone DLIB entry named AMACLIB. From the OUTDD operand, SMP/E determines the ddname of the operating system library and adds a SYSLIB subentry, MACLIB, to the DLIB entry. SMP/E uses this entry to determine the operating system library for subsequent modifications that specify an element's DISTLIB as AMACLIB.

### Copy Step C2

This copy step illustrates a selective copy of elements from a distribution library to an operating system library.

When a selective copy step is encountered, SMP/E assumes that the elements involved are **modules** and creates a target zone MOD entry for each of them. The module name is derived from the SELECT MEMBER statement; the distribution library for such modules is determined from the operand of the copy INDD statement; the load module name is simply the module name. In step C2, a MOD entry named LOADMODC is created; the distribution library is determined to be AOS14, and the LMOD is LOADMODC.

Finally, a target zone LMOD entry is created to represent the operating system library to which the module is copied. The load module name (and the target zone LMOD entry name) is the module name, LOADMODC. The operating system library (saved as a SYSLIB subentry) is determined from the operand of the copy OUTDD operand—in this case, LINKLIB. LMOD entries created from copy job steps do not have any link-edit attributes. Rather, an indicator (COPY) is set in the entry to inform SMP/E that the link-edit attributes must be obtained by examining the operating system library when the module must first be link-edited.

### **Assembly Step A1**

SMP/E creates an ASSEM entry with the name TESTMOD1, derived from the member name on the SYSPUNCH DD statement. This entry contains the assembler SYSIN data set.

A MAC entry named TESTMAC1 is created, because SMP/E detects the invocation of the macro in the assembler SYSIN. The macro entry created contains the name of the assembly as a GENASM subentry (for use by SMP/E in determining that this assembly should be performed if and when TESTMAC1 is updated).

### **Link-Edit Step L1**

This step shows the link-edit of the previous assembly. From the link-edit INCLUDE statement, SMP/E derives the data required to create a target zone MOD entry representing the module, TESTMOD1. The module name is determined from the member name operand on the INCLUDE statement, and the distribution library, SYSPUNCH, is determined from the INCLUDE statement's ddname.

Further, the link-edit defines the operating system library for a load module, LOADMOD1. A target zone LMOD entry is created from the link-edit NAME statement. It contains the ddname of the operating system library, derived from the link-edit SYSLMOD DD statement. In this case, the LMOD entry name is LOADMOD1, and the system library (saved as a SYSLIB subentry) is determined to be LPALIB. The load module attribute, RENT, is saved in the LMOD entry for use in subsequent link-edits of this load module; the parameters LET and LIST are not saved.

### **Link-Edit Step L2**

The second link-edit step defines two modules and one load module to SMP/E.

Modules TESTMOD2 and TESTMOD3 are defined by the link-edit INCLUDE statements; the distribution library for each of these is determined to be AOS12. A target zone MOD entry is created for each of these modules with a LMOD subentry naming the load module, LOADMODX, and a DLIB subentry, AOS12.

The operating system load module, LOADMODX, is represented by a target zone LMOD entry. This LMOD entry, as created, contains the operating system library, LINKLIB, as a SYSLIB subentry. No link-edit **attributes** are specified in this step; therefore, the STD indicator is set in the LMOD entry.

### Link-Edit Step L3

The third link-edit step shows an example of using the OPTIONS option. The OPTNAME DD statement allows SMP/E to process the PARM string, even though the options exceed the 100-character limit.

### Link-Edit Step L4

The fourth link-edit step shows an example containing a SYSLIB allocation. If CALLLIBS was specified on the JCLIN command or the ++JCLIN MCS, the low-level qualifiers of the data sets named in the SYSLIB allocation, PLIBASE and APPBASE, are saved as part of the CALLLIBS subentry list in LMOD entry LMOD04.

## Example 4: JCLIN for MAC and SRC Entries

Given the following JCLIN input:

```
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1    EXEC PGM=IEBCOPY
//MACDLIB  DD DSN=SYS1.MACDLIB,DISP=SHR
//MACTGT   DD DSN=SYS1.MACTGT,DISP=SHR
//SRCDLIB  DD DSN=SYS1.SRCDLIB,DISP=SHR
//SRCTGT   DD DSN=SYS1.SRCTGT,DISP=SHR
//SYSIN    DD *
COPY INDD=MACDLIB,OUTDD=MACTGT   TYPE=MAC
S      M=(MAC01,MAC02,MAC03)
S      M=(MAC04,MAC05)
COPY INDD=SRCDLIB,OUTDD=SRCTGT   TYPE=SRC
S      M=(SRC01,SRC02,SRC03)
S      M=(SRC04,SRC05)
/*
```

SMP/E creates the entries shown in Table 12:

Entry Type	Entry Name	DISTLIB	SYSLIB
MAC	MAC01	MACDLIB	MACTGT
MAC	MAC02	MACDLIB	MACTGT
MAC	MAC03	MACDLIB	MACTGT
MAC	MAC04	MACDLIB	MACTGT
MAC	MAC05	MACDLIB	MACTGT
SRC	SRC01	SRCDLIB	SRCTGT
SRC	SRC02	SRCDLIB	SRCTGT
SRC	SRC03	SRCDLIB	SRCTGT
SRC	SRC04	SRCDLIB	SRCTGT
SRC	SRC05	SRCDLIB	SRCTGT

### Example 5: JCLIN for an Assembler Step to Create a SRC Entry

Given the following JCLIN input:

```
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1    EXEC PGM=IEUASM
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//SYSPUNCH DD  DSN=&&PUNCH(SRCA),DISP=SHR
//SYSIN    DD  DSN=SYS1.ASRCLIB(SRCA),DISP=SHR
```

SMP/E creates a SRC entry named SRCA whose DISTLIB is ASRCLIB.

### Example 6: JCLIN for Load Modules Residing in a Hierarchical File System

A load module can reside in a hierarchical file system (HFS). To determine where the load module resides, SMP/E uses the following information, in addition to the usual JCL statements needed for load modules:

- The PATH operand on the SYSLIB or SYSLMOD statement associated with the load module. The PATH operand alerts SMP/E to the fact that the load module resides in an HFS; however, the PATH value specified is ignored.
- The LIBRARYDD comment statement immediately following the statement with the PATH operand. This comment statement specifies the ddname to be associated with the PATH value on the previous DD statement.
- The user-provided DDDEF entry whose name matches the ddname on the LIBRARYDD comment statement. The DDDEF entry specifies the directory portion of the pathname identified by the ddname. SMP/E uses the PATH value specified in the DDDEF entry to allocate the pathname, and does not check whether this value matches the PATH value specified on the SYSLIB or SYSLMOD DD statement associated with the LIBRARYDD comment.

Following are examples of job steps containing SYSLMOD and SYSLIB DD statements that use the PATH operand.

```

//STEP1 EXEC PGM=IEWBLINK,PARM='RENT,REUS'
1 //SYSLMOD DD PATH='/path_name1/'
2 //*LIBRARYDD=BPXLOAD1
//AOS12 DD DSN=SYS1.AOS12,DISP=SHR
//SYSLIN DD *
    INCLUDE AOS12(MOD00001)
    INCLUDE AOS12(MOD00002)
    ENTRY MOD00001
    NAME LMOD01(R)
/*
//STEP2 EXEC PGM=IEWBLINK,PARM='CALL,RENT,REUS'
//SYSLMOD DD PATH=SYS1.LINKLIB,DISP=OLD
//AOS12 DD DSN=SYS1.AOS12,DISP=SHR
3 //SYSLIB DD PATH='/path_calllib3/'
4 //*LIBRARYDD=BPXCALL3
4 // DD DSN=SYS1.PLIBASE,DISP=SHR
3 // DD PATH='/path_calllib4/'
4 //*LIBRARYDD=BPXCALL4
//SYSLIN DD *
    INCLUDE AOS12(MOD00005)
    INCLUDE AOS12(MOD00006)
    ENTRY MOD00005
    NAME LMOD03(R)
/*

```

- 1** Because the SYSLMOD statement specifies a PATH operand, SMP/E expects the next statement to be a LIBRARYDD comment statement.
- 2** Using the ddname on the LIBRARYDD comment, SMP/E updates the LMOD entry for LMOD01 to specify a SYSLIB value of BPXLOAD1. The user needs to provide a DDDEF entry for BPXLOAD1, specifying the appropriate pathname.
- 3** The SYSLIB DD statement is a concatenation of three DD statements. Two of the DD statements specify the PATH operand.
- 4** Using the ddnames on the LIBRARYDD comments and the low-level qualifier of the data set specified on the DSN operand, SMP/E updates the LMOD entry for LMOD03 to specify a CALLLIBS subentry list with the values BPXCALL3, PLIBASE, and BPXCALL4. The user needs to provide DDDEF entries for BPXCALL3 and BPXCALL4, specifying the appropriate pathnames. Likewise, the user needs to define PLIBASE with a DDDEF entry.

## Processing

This section discusses the following:

- Summary of JCLIN processing
- General JCLIN coding conventions
- Processing assembler steps
- Processing copy steps
- Processing link-edit steps
- Processing update steps
- Processing other utility steps

## Summary

When the JCLIN function of SMP/E is called, SMP/E begins to read the JCLIN input. It scans the JCL for job steps (that is, EXEC PGM=xxx or EXEC *procname*) containing information that SMP/E may need. The target or distribution zone is updated in the order in which the job steps are found in the JCLIN input.

### Reminder

The input for JCLIN must be free from all JCL errors and must conform to the SMP/E restrictions. Incorrect input can cause unpredictable errors.

SMP/E looks for certain program and procedure names that it recognizes as valid assembler, copy, link-edit, and update steps. It issues a warning message for other program and procedure names if it is not sure how to process them. For more information about programs and procedures not processed by JCLIN, see “Processing Other Utility Steps” on page 198.

If the JCLIN input contains program or procedure names that SMP/E does not recognize, these names can be passed to SMP/E by use of operands on the JCLIN control statement or ++JCLIN statement, as follows:

- ASM(PGM=*asmpgm*) or ASM(*asmproc*)
- COPY(PGM=*copypgm*) or COPY(*copyproc*)
- LKED(PGM=*lkedpgm*) or LKED(*lkedproc*)
- UPDATE(PGM=*updpgm*) or UPDATE(*updproc*)

**Note:** Only one additional program or procedure name can be specified for each utility. These names are in addition to the standard names built into SMP/E, and are lower in the search order than the built-in names. Therefore, if you specify **JCLIN UPDATE(PGM=ASMBLR)**, SMP/E still treats a job step specifying **PGM=ASMBLR** as an assembler.

The following sections describe JCLIN processing and coding conventions for each of the job steps SMP/E treats as significant. Remember that:

- Inline JCLIN processed at ACCEPT time updates the distribution zone.
- Inline JCLIN processed at APPLY time updates the target zone.
- The JCLIN command updates the target zone.

## General JCLIN Coding Conventions

SMP/E is not intended to support the wide variety of options and facilities supported in job control language (JCL); therefore, SMP/E has some rules as to how the JCL must be coded in order for SMP/E to process it correctly. If these conventions are not followed, the results are unpredictable: **JCLIN processing may not run successfully, and follow-on processing may be affected.** Certain conventions must be followed:

- Specify all information in uppercase characters (verbs as well as values). This is necessary to avoid syntax errors or incorrect results during SMP/E processing.

**Note:** This convention does not apply to values on the ALIAS statement. These values must be specified in the desired case (uppercase or mixed-case), because they are used as is.



- SMP/E does not recognize in-stream procedures. When SMP/E examines the JCLIN input, it looks only at JCL statements provided inline. That is, if you invoke any cataloged procedures in the JCLIN input, SMP/E does not expand those procedures in order to access any required DD statements. If you use cataloged procedures, they must follow the same conventions as the IBM-supplied system generation cataloged procedures (ASMS for assemblies and LINKS for link-edits). The conventions are described in later sections.
- The ddname for a distribution library or target library should match the lowest-level qualifier of the data set name. This is not a requirement, but rather a recommendation to help you specify the data sets used by SMP/E. During processing, a data set is referred to by its ddname, not the complete data set name. If the ddname and the lowest-level qualifier DSN are kept the same, correlating these names may be easier for you. For example, the DD statement for SYS1.LINKLIB can be specified as:

```
//LINKLIB DD DSN=SYS1.LINKLIB,DISP=SHR
```

This convention for correlating ddnames and data set names is used by IBM when it distributes functions or PTFs. Therefore, you should avoid changing the ddnames of libraries used for elements provided by IBM. If you change these ddnames in the element entries, the ddnames for these elements in subsequent SYSMODs may not match the ddnames in the entries, and SMP/E may not be able to process the SYSMODs.

- Concatenated data sets must not be used for input. For example, a link-edit step has an INCLUDE statement in its input that says:

```
INCLUDE DD1(MODA,MODB,MODC)
```

and has the following DD statements:

```
//DD1 DD DSN=SYS1.USRDLIB1,DISP=SHR
// DD DSN=SYS1.USRDLIB2,DISP=SHR
// DD DSN=SYS1.USRDLIB3,DISP=SHR
```

In this case, SMP/E is able to process the JCL, but the updates to the zone being processed are not sufficient to enable service to be installed. There are two problems with this example:

- The ddnames are not the same as the lowest-level qualifier of the DSN.
- The data sets are concatenated.

The following examples show the correct format for the INCLUDE statements and DD statements:

```
INCLUDE USRDLIB1(MODA)
INCLUDE USRDLIB2(MOdB)
INCLUDE USRDLIB3(MODC)
```

and

```
//USRDLIB1 DD DSN=SYS1.USRDLIB1,DISP=SHR
//USRDLIB2 DD DSN=SYS1.USRDLIB2,DISP=SHR
//USRDLIB3 DD DSN=SYS1.USRDLIB3,DISP=SHR
```

- A continued utility control statement should be used only when absolutely necessary. Although SMP/E attempts to support all existing formats of the utility control statements, it cannot completely duplicate the syntax checking of the utility. The safe method is to use the simplest format of the utility control statement.

For example, rather than code:

```
INCLUDE AOS12(GIMMPDRV,GIMMPDR1,GIMMPDR2,          X
             GIMMPDC1,GIMMPDC2)
```

a better (and safer) method is:

```
INCLUDE AOS12(GIMMPDRV,GIMMPDR1,GIMMPDR2)
INCLUDE AOS12(GIMMPDC1,GIMMPDC2)
```

The link-edit utility accepts both formats, but the second example is the safe format.

**Note:** Generally, SMP/E does **not** require a continuation character in column 72. However, if a link-edit control statement is to be continued to the next statement, a nonblank continuation character **must** be placed in column 72. This is necessary to avoid syntax errors or incorrect results during SMP/E processing.

- The DD statement identifying the input control statements for a utility **must be** the last DD statement in that job step.

For example:

```
//STEP1 EXEC PGM=IEWL
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12 DD DSN=SYS1.AOS12.DISP=SHR
//SYSLIN DD *
INCLUDE AOS12(GIMMPDRV,GIMMPDR1,GIMMPDR2,...)
ENTRY GIMMPDRV
SETCODE AC(1)
NAME GIMMPP(R)
/*
```

The above is valid, but the following is **not** valid.

```
//STEP1 EXEC PGM=IEWL
//SYSLIN DD *
INCLUDE AOS12(GIMMPDRV,GIMMPDR1,GIMMPDR2,...)
ENTRY GIMMPDRV
SETCODE AC(1)
NAME GIMSMP(R)
/*
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12 DD DSN=SYS1.AOS12.DISP=SHR
```

For each step encountered, SMP/E saves all the JCLIN records in a work area until the first non-JCL (that is, first statement after the SYSIN DD statement or the SYSLIN DD statement) is found. The saved records are then searched for any information that must be obtained from the JCL. If a JCL statement follows the utility input DD statement, that JCL statement will not be in the work area when the search performed, and SMP/E might not be able to capture the necessary data.

**Note:** SMP/E has no column limitations for operands beyond the normal JCL rules.

## Processing Assembler Steps

Assembler steps are identified by one of the following:

- EXEC PGM=IFOX00
- EXEC PGM=IEUASM
- EXEC PGM=IEV90
- EXEC PGM=ASMBLR
- EXEC ASMS
- EXEC PGM=*asmpgm* [if the JCLIN specified ASM(PGM=*asmpgm*)]
- EXEC *asmproc* [if the JCLIN specified ASM(*asmproc*)]

Either ASSEM or SRC entries are built, depending on where the assembler input is obtained.

- ASSEM entries are built when the assembler input is inline (the SYSIN DD statement does not point to another data set).
- SRC entries are built when the SYSIN DD statement points to a member of a partitioned data set.

### Building ASSEM Entries

As the assembler step is processed, every assembler control statement [that is, every card from the SYSIN DD \* statement until the end of input (/ \* or //)] is written as an ASSEM entry.

The name of the ASSEM entry is determined by looking at the JCL for the assembly step. If the step is of the EXEC PGM=*name* type, SMP/E looks for any of the following statements to find the member name for the DSN or DSNAME library:

```
//SYSPUNCH DD DSN=...
```

or

```
//SYSGO DD DSN=...
```

or

```
//SYSLIN DD DSN=...
```

These are examples of the statements SMP/E looks for:

```
//SYSPUNCH DD DSN=high.next.low(member)
```

or

```
//SYSPUNCH DD DSNAME=high.next.low(member)
```

The SYSPUNCH, SYSGO, or SYSLIN DD statement must point to a PDS, and the member must be specified.

If the step is of the EXEC *asmproc* type, SMP/E looks for the MOD=*name* operand of the PROC statement. If it is found, that name is used as the ASSEM entry name. If not, one of three DD statements must be present—SYSPUNCH, SYSGO, or SYSLIN DD—and the ASSEM entry name is obtained from the DD statement. If any ASSEM entry previously existed, the current set of assembler input completely replaces the previously saved assembler input.

### Building SRC Entries

If the SYSIN DD statement points to a member of a partitioned data set, SMP/E assumes that the member is a source. It then creates a SRC entry whose name is the same as the member name on the SYSIN DD statement. The name of the DISTLIB for the SRC entry is the low-level identifier of the data set name on the SYSIN DD statement.

### Building MAC Entries

In the process of reading and writing inline assembler input and building an ASSEM entry, the operation field (operation codes) of each assembler statement is scanned for macro invocations. If column 1 of the assembler statement is blank, SMP/E considers the first character string found to be the operation code. If column 1 is not blank, the second character string found is the operation code. SMP/E recognizes the following as macros:

- An operation code of six to eight characters in length
- The operand of the assembler COPY operation

For each operation code found, SMP/E determines whether it is a macro invocation or an assembler instruction. It does so by providing a PARMLIB member (GIMOPCDE) identifying character strings that are to be treated as operation codes or macros. This member identifies all the operation codes in the *S/370 Reference Summary*, GX20-1850, as well as all the assembler instructions supported by OS/VS Assembler H. The user can build an optional PARMLIB member identifying additional operation codes or macros. SMP/E builds a table using the PARMLIB member (GIMOPCDE) and any optional PARMLIB member specified in either the JCLIN command or the ++JCLIN statements. Statements found in the optional PARMLIB member, identifying character strings also specified in GIMOPCDE, take precedence. The operation fields (operation codes) of the assembler input are scanned during JCLIN processing and are compared to the table built from the PARMLIB members; if the character is not found, or if it is found but specified as **TYPE=MACRO**, the character string is treated as a macro name and is processed as a macro.

If PARMLIB is not available or the specified member is not found, SMP/E determines whether the character string is an operation code or macro by using default logic, which is that any character string less than six characters long is an operation code. In this case, a message is issued for each character string found, indicating how SMP/E treated it, why, and what assembly it was found in.

For a description of the format of the PARMLIB member control cards, see Chapter 33.

For each macro found in the assembly input, SMP/E does the following:

- Locates the MAC entry in the zone being processed. If the entry is found, it is modified. If the entry is not found, a new entry is created.
- Updates the MAC entry by adding the name of the assembly module (as described above) as a GENASM subentry. This now means that each macro used during the assembly includes a reference to the target zone ASSEM entry. Therefore, when a SYSMOD is processed that modifies that macro, SMP/E knows what target zone ASSEM entries should be reassembled in order to include the new macro.

**Note:** After JCLIN processing that creates a new MAC entry, the only data present in the MAC entry is the set of GENASM subentries. Additional data, such as the distribution library, is added to the MAC entry during the installation of the SYSMOD supplying the actual macro.

## Processing Copy Steps

Copy steps are identified by one of the following:

- EXEC PGM=IEBCOPY
- EXEC PGM=*copypgm* [if the JCLIN specified COPY(PGM=*copypgm*)]
- EXEC *copyproc* [if the JCLIN specified COPY(*copyproc*)]

When a copy step is encountered, SMP/E searches through the JCL looking for the SYSIN DD \* statement. All records from the SYSIN DD \* statement to the end of input (/ \* or //) are assumed to be the copy input control statements.

**Note:** This requires the copy input to be **inline**, not pointing to another data set.

When scanning the copy input, SMP/E assumes the ddnames of the statement are the same as the lowest-level qualifier of the data set referred to. If the COPY statement is set up without this convention, SMP/E generates incorrect DLIB and SYSLIB names in the MOD and LMOD entries.

By scanning the IEBCOPY control statements, SMP/E knows which members of the DLIB are being copied (from the IEBCOPY SELECT MEMBER statements), from which DLIB they are being copied (the IEBCOPY INDD ddname), and to which target libraries they are going (the IEBCOPY OUTDD ddname).

The members can be macros, modules, source elements, or data elements. SMP/E has no way of determining the type from the standard IEBCOPY control statements. To find this information, SMP/E looks for the TYPE comment on the COPY INDD=*ddname*,OUTDD=*ddname* control statement. The format of the TYPE comment on the COPY statement is:

```
COPY INDD=ddname,OUTDD=ddname TYPE=xxxx
```

where:

*xxxx*

is MOD, MAC, SRC, or DATA, which indicates the type of element in the distribution library.

## Notes:

1. If the **TYPE=xxxx** parameter is not specified, the default is **TYPE=MOD**.
2. **DATA** is used for data elements. If **DATA** is specified, SMP/E skips to the next **COPY** control statement without creating any entries for the data elements.

There must be at least one blank between the copy statement and the comment. **TYPE=xxxx** must be the first character string in the comment, and no blanks must precede or follow the "=" sign. This information is used only if a **SELECT** statement follows the **COPY** statement.

The **SELECT** statement can name either the member to be copied or an alias name for a member. The format of these **SELECT** statements is:

```
SELECT MEMBER=member  
SELECT MEMBER=alias           ALIAS OF member
```

The first **SELECT** statement identifies a member to be copied. The second identifies an alias and names the associated member in the comment portion of the statement.

**Note:** A **SELECT** statement identifying an alias can specify only one name on the **MEMBER** operand and cannot specify **RENAME** or **REPLACE**.

## Building DLIB Entries

If a **COPY** statement is followed either by another **COPY** statement or by an **EXCLUDE** statement, SMP/E assumes that the **INDD** library is totally copied to the **OUTDD** library. Because SMP/E does not look at either the **DLIB** or target library during JCLIN processing, it has no way of knowing what elements are contained in the **INDD** library. Therefore, a **DLIB** entry with the same name as the **INDD** ddname is created. This indicates that the entire library has been copied to the library specified by the **OUTDD** ddname. If the **INDD** ddname **DLIB** entry already exists, SMP/E adds the **OUTDD** ddname to the **DLIB** entry. SMP/E records these ddnames in alphabetical order. A **DLIB** entry can have at most two **SYSLIB** subentries. If the **DLIB** entry already has two **SYSLIB** subentries, SMP/E replaces the second **SYSLIB** ddname with the ddname specified on the **OUTDD** copy statement operand.

Later, whenever a **SYSMOD** is processed during **APPLY** or **ACCEPT**, SMP/E can check the **DISTLIB** ddname for each of the **SYSMOD**'s elements to determine whether that element belonged to a totally copied distribution library. This is how **DLIB** entries are used during **APPLY** and **ACCEPT** processing:

- For **++MOD** statements, SMP/E can construct a **MOD** entry indicating that the module has been copied and that the load module name is the same as the module name. In addition, an **LMOD** entry can be built, indicating a copy with a **SYSLIB** equal to the **SYSLIB** value in the **DLIB** entry.
- For **++MAC**, **++MACUPD**, **++SRC**, **++SRCUPD**, and data element **MCSs**, SMP/E can determine that the element is part of a totally copied library, and can then fill in the **SYSLIB** field in the appropriate element entry with the **SYSLIB** ddname saved in the **DLIB** entry. If there are two **SYSLIB** ddnames in the **DLIB** entry, SMP/E uses the second value.

**Note:** SMP/E treats copies with exclude as total copies. It does not retain information about which elements were excluded during the creation of the initial library.

### Building MOD and LMOD Entries

If the COPY statement is followed by a SELECT statement and either specifies **TYPE=MOD** or lets TYPE default to MOD, SMP/E builds MOD and LMOD entries for the elements specified in the SELECT statement.

The COPY and SELECT statements specify the members of the DLIB that are being copied (the IEBCOPY select member statements), aliases for these members (the IEBCOPY select member statements with alias comments), the DLIB they are being copied from (the IEBCOPY INDD ddname), the target libraries they are going to (the IEBCOPY OUTDD ddname), and the name of the load module in the target library (the IEBCOPY select member statements).

For each member in the SELECT statement list, SMP/E builds:

- A MOD entry with the same name as the selected element, with a DISTLIB value equal to the INDD operand value and an LMOD subentry with the same name as the copied element (unless the rename function of the copy select statement was used, in which case the LMOD subentry name is as specified in the RENAME operand).

If a SELECT statement names an alias as the member, that name is added as a TALIAS value in the MOD entry for the module named in the SELECT statement comment.

- An LMOD entry with the same name as the selected element (unless the rename function of the copy select statement was used, in which case the LMOD name is as specified in the RENAME operand) is created, indicating that the load module has been copied and that its SYSLIB is equal to the ddname specified in the OUTDD COPY statement operand.

An LMOD entry can have at most two SYSLIB subentries. If the LMOD entry already has two SYSLIB subentries, SMP/E replaces the second SYSLIB ddname with the ddname specified on the OUTDD COPY statement operand.

**Note:** Because SMP/E only looks at the JCLIN input and has no knowledge about the DLIBs or target libraries built into it, SMP/E cannot determine what types of libraries are being copied. SMP/E assumes all libraries that are selectively copied are load libraries; therefore, MOD and LMOD entries are built for all selectively copied elements. Thus, if a macro or source DLIB is selectively copied, SMP/E **does not** build MAC entries and SRC entries, but builds extraneous MOD and LMOD entries.

### Building MAC Entries

If the COPY statement is followed by a SELECT statement and specifies **TYPE=MAC**, SMP/E builds MAC entries for the elements specified in the SELECT statement. SMP/E determines the macro name from the SELECT statement, the macro DISTLIB from the INDD=*ddname* parameter on the COPY statement, and the macro SYSLIB from the OUTDD=*ddname* parameter on the COPY statement.

If no entry exists for a macro, SMP/E creates one using the DISTLIB and SYSLIB information obtained from the COPY statements.

If an entry already exists for a macro, SMP/E compares the DISTLIB and SYSLIB subentries in the existing entry to those obtained from the COPY statements. If they are different, SMP/E updates the macro entry with the values from the COPY statement.

If a SELECT statement names an alias as the member, that name is added as a MALIAS value in the MAC entry for the macro named in the SELECT statement comment.

SMP/E does not support the RENAME option of the COPY statement.

### Building SRC Entries

If the COPY statement is followed by a SELECT statement and specifies **TYPE=SRC**, SMP/E builds SRC entries for the elements specified in the SELECT statement. SMP/E determines the source name from the SELECT statement, the source DISTLIB from the **INDD=ddname** parameter on the COPY statement, and the source SYSLIB from the **OUTDD=ddname** parameter on the COPY statement.

If no entry exists for the source, SMP/E creates one using the DISTLIB and SYSLIB information from the COPY statements. If a source subentry already exists, SMP/E compares the DISTLIB and SYSLIB subentries in the existing entry to those obtained from the COPY statements. If they are different, SMP/E updates the source entry with the values from the COPY statement.

SMP/E does not support the RENAME option of the COPY statement.

## Processing Link-Edit Steps

Link-edit steps are identified by one of the following:

- EXEC PGM=IEWL
- EXEC PGM=HEWL
- EXEC PGM=IEWBLINK
- EXEC PGM=*lkedpgm* [if the JCLIN specified LKED(PGM=*lkedpgm*)]
- EXEC LINKS
- EXEC *lkedproc* [if the JCLIN specified LKED(*lkedproc*)]

When SMP/E encounters a link-edit step, it reads through the JCL control statements looking for the SYSLIN DD statement containing the link-edit control card input. All records from the SYSLIN DD statement to the end of input (*/\** or *//*) are assumed to be link-edit utility control statements.

- These records **must be** control statements, not object modules.
- All link-edit control statements must start in or after column 2.
- The SYSLIN input **must be** inline. It cannot point to a member of another data set, because SMP/E then could not analyze the data.

**Note:** When SMP/E detects the */\*SMPLTS* comment during JCLIN processing, it skips all the steps in the SMPLTS job.

Link-edit steps must not be sensitive to the order of execution of other link-edit steps either for the same FMID or for another, conditionally coexistent FMID. No elements to be included in a link-edit step should be derived from the output of another link-edit step. Also, if multiple load modules and target libraries are



involved, SMP/E organizes the link-edit steps for the most efficient invocations of the link-edit utility (which might not be in the same order as in the JCLIN data).

For example, suppose a product consists of a base function and a dependent function. The dependent function conditionally coexists with the base function; it can be installed with the base function, but is not a prerequisite for the base function. The base function must have its own JCLIN data that completely describes the elements it contains, because a user may choose to install the functions together or separately. If the base function is installed separately, its JCLIN data cannot contain a link-edit step that includes elements from the dependent function, because those elements are not yet available.

When scanning the input, SMP/E looks for and performs special processing for selected DD statements and link-edit control statements.

- Some DD statements and link-edit statements are processed to create MOD and LMOD entries.
- Others are not processed, but are saved as is in the LMOD entry so they can be passed to the link-edit utility when the load module needs to be relinked.

**Note:** These statements may be processed by the link-edit utility after they are saved in the LMOD entry by the command being run. For example, if the JCLIN data is being processed by the APPLY command, these statements may be processed by the link-edit utility as part of installing the module. On the other hand, if the JCLIN data is being processed by the JCLIN command, these statements are saved in the LMOD entry and are passed to the link-edit utility only when the module is link-edited again at some future date.

For more information about link-edit utility rules, see the link-edit utility manual for your operating system.

- Some DD statements should not be specified at all as JCLIN data. They are not processed by the JCLIN command, but they are saved in the LMOD entry and can produce undesirable results when they are processed later.

Table 13 on page 190 and Table 14 on page 190 show a summary of this information. They are followed by more detailed guides.

#### Reminder

JCLIN by itself does not force SMP/E to link-edit a load module. The JCLIN only causes SMP/E to update the CSI entries with the information in the JCLIN. To force a link-edit, you must also supply the appropriate element statements (++MOD).

For example, to add an alias to a load module, you must supply JCLIN and a ++MOD statement for a module contained in the load module. The JCLIN updates the LMOD entry with the new link-edit control statements, and the ++MOD statement tells SMP/E to perform the link-edit.

*Table 13. Summary of How DD Statements Are Processed as JCLIN Data*

Statement	Processed to Create Entries	Saved Only as ++LMODIN Data in the LMOD Entry	Should Not Be Specified in JCLIN Data
SYSLIB	Yes (See note)		
SYSLMOD	Yes		

**Note:** A SYSLIB DD statement is processed only if CALLLIBS was specified on the JCLIN command or the ++JCLIN MCS.

*Table 14. Summary of How Link-Edit Control Statements Are Processed as JCLIN Data*

Statement	Processed to Create Entries	Saved Only as ++LMODIN Data in the LMOD Entry	Should Not Be Specified in JCLIN Data
ENTRY		Yes	
EXPAND			Do not specify in JCLIN data
IDENTIFY			Do not specify in JCLIN data
INCLUDE	Yes		
INSERT and OVERLAY		Yes	
LIBRARY			Do not specify in JCLIN data (See note)
NAME	Yes		
ORDER		Yes	
All other statements except comments		Yes	

**Note:** Do not use LIBRARY statements to specify additional automatic call libraries.

**ALIAS statement**

To ensure that SMP/E can process your link-edit ALIAS control statements, you must address the following considerations:

- **General considerations**

- An ALIAS control statement can span any number of 80-byte records.

**Note:** If you assign a load module residing in a PDSE an alias value greater than eight characters, you cannot later use the ++DELETE statement to delete that alias value (and not the associated load module). To delete such an alias value without deleting the load module, you need to resupply JCLIN to define the load module without providing an ALIAS statement for the alias value to be deleted. Make sure to also include a ++MOD

statement for a module in the load module to force SMP/E to relink the load module.

- Column 1 of all 80-byte records composing an ALIAS control statement must contain a blank (X'40').
- The data for the first 80-byte record of an ALIAS control statement must start in column 2 or later and end anywhere up to and including column 71.
- The control statement type (ALIAS) must be followed by at least 1 blank (X'40').
- The control statement type (ALIAS) must be in uppercase.
- Columns 73 through 80 of an 80-byte record are ignored.
- An alias value can be from one to 64 characters.
- An alias value can be composed of characters in the range X'41' through X'FE'.

**Note:** Although the binder also accepts characters X'0E' (shift-out character) and X'0F' (shift-in character), SMP/E does not accept them.

- An alias value can be enclosed in single apostrophes. It must be enclosed in single apostrophes in the following cases:
  - It contains a character other than uppercase alphabetic, numeric, national (\$, #, or @), slash, plus, hyphen, period, and ampersand. Here is an example:

```

      1      2      3      4      5      6      7      8
-----+-----+-----+-----+-----+-----+-----+-----+-----0
ALIAS 'This_alias_contains_special_characters!!!!'
```

- It is continued to another 80-byte record of the control statement. Here is an example:

```

      1      2      3      4      5      6      7      8
-----+-----+-----+-----+-----+-----+-----+-----0
ALIAS      'This_is_a_very_long_value_that_is_continued_to_the_next*
_card!'
```

- If an apostrophe is part of the alias value (not a delimiter), two apostrophes need to be specified in the appropriate location in the alias value. These two apostrophes count as two characters in the 64-character limit for an alias value. Here is an example:

```

      1      2      3      4      5      6      7      8
-----+-----+-----+-----+-----+-----+-----+-----0
ALIAS 'It''s_the_quote_that_makes_apostrophes_necessary.'
```

- The single apostrophes used to enclose an alias value do not count as part of the 64-character limit for an alias value. For example, the alias value in the following example contains 10 characters:

```

      1      2      3      4      5      6      7      8
-----+-----+-----+-----+-----+-----+-----+-----0
ALIAS 'Only_ten!!!'
```

- SMP/E uses the alias value exactly as is. SMP/E does not try to enforce any rules the binder may be using as a result of the CASE execution parameter.

- **Continuation records**

- Column 72 of a given 80-byte record must be a nonblank character if the control statement is continued onto the next 80-byte record. The character in column 72 denotes only continuation and is never part of an alias value.
- The data for continuation records (80-byte records 2 through *n* of an ALIAS control statement) can start in column 2 or later and end anywhere up to and including column 71 (for example, if multiple aliases are being specified).

The data for a continuation record must start in column 2 if it is part of an alias value that is being continued from the previous 80-byte record. An alias value that is continued from one 80-byte record to another 80-byte record must do all of the following:

- Be enclosed in single apostrophes
- Extend through column 71 of the first 80-byte record
- Start in column 2 of the next 80-byte record
- Have a nonblank continuation character in column 72

Here is an example:

```

      1      2      3      4      5      6      7      8
-----0-----0-----0-----0-----0-----0-----0-----0
ALIAS      'This_is_a_very_long_value_that_is_continued_to_the_next*
_card!'
```

- **Entry points**

A new format of the ALIAS statement supported for the binder allows an alternative entry point to be specified into a load module. If this new format is used, each alias name with an associated entry point must be specified on its own 80-byte record, with a separate ALIAS statement; no other aliases should be specified on that statement. If multiple alias values of this format are specified on a single ALIAS control statement, only the first is recognized; the rest are ignored.

**Note:** When this form of the ALIAS control statement is used, the alias value cannot be 64 characters long, because SMP/E requires the statement to be complete on one 80-byte record. When this form of the ALIAS control statement is used, the maximum length for an alias value is 61 characters.

Suppose that a load module has the following aliases: ALA1, ALA2, ALA3, and ALA4. ALA1 and ALA2 are associated with entry point names ENTRYPT1 and ENTRYPT2, respectively.

- Here are examples of how the aliases should be specified:

```

      1      2      3      4      5      6      7      8
-----0-----0-----0-----0-----0-----0-----0-----0
ALIAS ALA1(ENTRYPT1)
ALIAS ALA2(ENTRYPT2)
ALIAS ALA3
ALIAS ALA4
or
ALIAS ALA3,ALA4
```

- Here are examples of how the aliases should *not* be specified:

```

      1      2      3      4      5      6      7      8
-----+-----+-----+-----+-----+-----+-----+-----+-----+
ALIAS ALA1(ENTRYPT1),ALA2(ENTRYPT2),ALA3,ALA4
or
ALIAS ALA1(ENTRYPT1),ALA3
ALIAS ALA2(ENTRYPT2),ALA4

```

- **Multiple aliases**

- Multiple alias values can be specified on a single ALIAS control statement as long as they are not in the form alias(entrypoint). Multiple alias values must be separated by commas (“,”). Here is an example:

```

      1      2      3      4      5      6      7      8
-----+-----+-----+-----+-----+-----+-----+-----+
ALIAS ALIAS1,ALIAS2,ALIAS3,ALIAS4

```

- Multiple alias values can span multiple 80-byte records. When this occurs, there must be a nonblank character in column 72, and one of the following must be true:

- The last alias value on the 80-byte record that is being continued must be followed by a comma and one or more blanks (“, ...”). Here is an example:

```

      1      2      3      4      5      6      7      8
-----+-----+-----+-----+-----+-----+-----+-----+
ALIAS ALIAS1,ALIAS2,
ALIAS3,ALIAS4

```

- The last alias value on the 80-byte record that is being continued must be followed by a comma (“,”) in column 71. Here is an example:

```

      1      2      3      4      5      6      7      8
-----+-----+-----+-----+-----+-----+-----+-----+
ALIAS ALIAS1,ALIAS2,'A_relatively_long_ALIAS_but_not_quite_64_chars.',*
ALIAS4,ALIAS5

```

- The last alias value on the 80-byte record that is being continued can be enclosed in single apostrophes such that part of the alias value appears on the current 80-byte record and part appears on the next 80-byte record (see the rules for continuation records). Here is an example:

```

      1      2      3      4      5      6      7      8
-----+-----+-----+-----+-----+-----+-----+-----+
ALIAS ALIAS1,ALIAS2,'A_relatively_long_ALIAS.',ALIAS4,'Not_too_long_bu*
t_wraps.',ALIAS5,ALIAS6

```

- If a blank (X'40') follows an alias value, SMP/E assumes there are no more alias values for the current ALIAS control statement.

### CHANGE statement

CHANGE statements are saved in the LMOD entry and are associated with the DLIB module name found on the next INCLUDE statement in the JCL. If the same INCLUDE statement is processed later by JCLIN, CHANGE statements in the LMOD entry associated with this INCLUDE statement are deleted and then replaced by any associated CHANGE statements in the latest job. CHANGE statements are passed to the linkage editor only when the associated DLIB module is to be replaced in the load module.

A function SYSMOD must not contain a CHANGE statement in a link-edit step unless PTFs can be built without an IDENTIFY statement for the changed CSECT.

**Note:** RESTORE processing is limited for a SYSMOD that uses the CHANGE statement in inline JCLIN. When such a SYSMOD is applied, the existing LMOD entry (if any) is first backed up on the SMPSCDS and is updated with the inline JCLIN containing the CHANGE statement. When that SYSMOD is restored, the backup copy of the LMOD entry (which does not have the updates from the CHANGE statement) replaces the target zone LMOD entry, and the information from the CHANGE statement is lost. Modules whose names were changed by the inline JCLIN remain in the load module under their changed names.

**ENTRY statement**

Each load module consisting of more than one distribution library module must have an ENTRY statement; otherwise, the entry point of the load module changes each time the load module is relinked by SMP/E.

**EXPAND statement**

EXPAND statements should not be used in JCLIN data, because they would be saved in the LMOD entry and would cause the load module to be expanded each time it is link-edited. This is not always desirable. However, the EXPAND statement is allowed as input for the ++ZAP MCS. See "Usage Notes" on page 599 for more information.

**IDENTIFY statement**

IDENTIFY statements should not be used in JCLIN data. They are produced as part of servicing a module. If found in the JCLIN, they are stored in the LMOD entry and can result in incorrect data being stored during the application of service.

**INCLUDE *ddname(member,member...)* statement**

Each module in the load module must be specified as a member name on an INCLUDE statement.

The member names are assumed to be modules existing in distribution library *ddname*. SMP/E builds MOD entries for each member name specified and sets the DISTLIB value in each MOD entry to *ddname*. (An exception to this is when the *ddname* is SYSLMOD. In that case, no MOD entry is built for the INCLUDE statement.) SMP/E does not refer to the *ddname* DD statement to determine the actual library referred to. Therefore, all *ddnames* specified on INCLUDE statements must be the actual *ddnames* assigned to the products.

The INCLUDE statements are not saved in the LMOD entry, because they are not necessary when the load module is link-edited. All link-edits requested by SMP/E are CSECT-replaces; the load module is built from the new version of the updated CSECT and the existing load module from the target library.

The *ddnames* *SYSPUNCH* and *SMPOBJ* are reserved for inclusions of object decks produced by assembly steps that are not to be link-edited to a distribution library during ACCEPT processing. In both cases, the name stored in the MOD entry's DISTLIB subentry is SYSPUNCH.

**INSERT and OVERLAY statements**

If a load module is to be linked in overlay structure, you must supply an INSERT control statement for each CSECT in the load module, including INSERT statements for those CSECTs within the root segment. It is not sufficient to properly place the INCLUDE and OVERLAY control statements.

**LIBRARY statement**

Normally, LIBRARY statements should not be used in JCLIN data. An exception is when the CALLLIBS operand is specified on the JCLIN command or ++JCLIN MCS, or when /\*CALLLIBS=YES is encountered after a job card preceding a link-edit step. JCLIN processing then allows for the LIBRARY statement to be used to specify those modules (external references) that are to be excluded from the automatic library search during the following:

- The current linkage editor job step (restricted no-call function)
- Any subsequent linkage editor job step (never-call function)

A LIBRARY statement should be used only if a SYSLIB DD statement is also used. It should not be used to specify additional automatic call libraries; the SYSLIB DD statement should be used instead.

**NAME *lmodname*(R) statement**

When SMP/E encounters either the NAME control statement or the end of input with no NAME statement, SMP/E builds an LMOD entry. How SMP/E determines the name of the LMOD depends on the JCL being scanned:

- If the NAME statement is found, SMP/E gets the LMOD name is from the *lmodname* field of the NAME statement.
- If no NAME statement is found and a SYSLMOD DD statement is present, SMP/E gets the LMOD name from the member name of the data set specified. If no member name is specified, SMP/E issues an error message identifying the JOBNAME and STEPNAME and the reason for the error.
- If no NAME and SYSLMOD DD statements are found, SMP/E searches for the MOD=*name* operand in the JCL and uses that name as the LMOD name. If no MOD=*name* operand is found, SMP/E issues an error message.

**ORDER statement**

If a specific order of CSECTs within a load module is necessary, ORDER statements are required to define the load module structure. Simply ordering the INCLUDE statements is not sufficient, because SMP/E does CSECT replacements when relinking the load module and, therefore, changes the order of the CSECTs.

**REPLACE statement**

REPLACE statements are saved in the LMOD entry and are associated with the DLIB module name found on the next INCLUDE statement in the JCL. If the same INCLUDE statement is processed later by JCLIN, REPLACE statements already in the LMOD entry associated with this INCLUDE statement are deleted and replaced by any associated REPLACE statements in the latest job. REPLACE statements are passed to the linkage editor only when the associated DLIB module is to be replaced in the load module.

**SYSLIB DD statement**

Normally, SYSLIB DD statements should not be used in JCLIN data. However, they can be used with SMP/E Release 8 or later for load modules needing to implicitly include modules from other products. Such load modules are commonly used by products that:

- Are written in a high-level language and, as a result, include modules from libraries (such as compiler libraries) that are owned by a different product

- Make use of a callable-services interface provided by another product
- Need to include stub routines or interface modules from different products that may reside in other zones

For such load modules, the SYSLIB DD statement should specify all the automatic call libraries SMP/E is to use when linking the load module. (These libraries should be target libraries.) The low-level qualifier of each data set specified in the SYSLIB concatenation is saved as a CALLLIBS subentry for the associated load module. For SMP/E to link implicitly-included modules from these libraries, the user must provide DDDEF entries for the libraries in the zone containing the LMOD entry.

**Note:** Zones containing LMOD entries with CALLLIBS subentries cannot be processed by SMP/E Release 7 or earlier.

SYSLIB DD statements are processed only if the CALLLIBS operand is specified on the JCLIN command or ++JCLIN MCS, or if `/*CALLLIBS=YES` is encountered after a job card preceding a link-edit step. If the CALLLIBS operand or the CALLLIBS comment is not specified, SMP/E ignores any SYSLIB DD statements it encounters.

**Including Pathnames in a SYSLIB Concatenation:** A DD statement in a SYSLIB concatenation can include the PATH operand to specify a pathname as an automatic call library. A LIBRARYDD comment statement must immediately follow this DD statement and specify the ddname to be associated with that pathname. SMP/E saves the ddname specified on the LIBRARYDD comment as part of the CALLLIBS list in the LMOD entry being updated or created. For an example, see “Example 6: JCLIN for Load Modules Residing in a Hierarchical File System” on page 178.

**Notes:**

1. If a DD statement in the concatenation comes between the DD statement specifying the PATH operand and the LIBRARYDD comment, the misplaced DD statement is ignored.
2. If the DD statement specifying the PATH operand is followed by a JCL statement other than a LIBRARYDD comment or a continuation DD statement for the SYSLIB concatenation, the LMOD entry is not updated or created. In addition, if the JCLIN was specified in a SYSMOD (instead of being processed by the JCLIN command), processing for that SYSMOD fails.

**SYSLMOD DD statement**

SMP/E uses either the SYSLMOD DD statement or the NAME statement to determine the target library for the load module, as follows:

- If a SYSLMOD DD statement is present, SMP/E determines the target library ddname by using the lowest-level qualifier of the data set name specified in the SYSLMOD DD statement.
- If no SYSLMOD DD statement is present, SMP/E determines the name by looking at the `NAME=dsname` option on the procedure statement. The ddname used is the lowest-level qualifier of the data set name specified in the NAME option.
- If no SYSLMOD DD statement or `NAME=dsname` value is found, SMP/E issues an error message.



The ddname of the target library is saved as the SYSLIB value in the LMOD entry for the load module.

A SYSLMOD DD statement can include the PATH operand to specify a pathname for installing a load module in a hierarchical file system. A LIBRARYDD comment statement must immediately follow this DD statement and specify the ddname to be associated with that pathname. SMP/E saves the ddname specified on the LIBRARYDD comment as a SYSLIB subentry in the LMOD entry being updated or created. For an example, see “Example 6: JCLIN for Load Modules Residing in a Hierarchical File System” on page 178.

**Notes:**

1. If the DD statement specifying the PATH operand is followed by a JCL statement other than a LIBRARYDD comment, the LMOD entry is not updated or created. In addition, if the JCLIN was specified in a SYSMOD (instead of being processed by the JCLIN command), processing for that SYSMOD fails.
2. An LMOD entry can have at most two SYSLIB subentries. If the LMOD entry already contains two SYSLIB subentries, SMP/E replaces the second SYSLIB ddname with the ddname found on the SYSLMOD DD statement, the NAME=*dsname* option, or the LIBRARYDD comment statement.

**All other statements found in link-edit input**

All other link-edit control statements found are saved in the LMOD entry in the order they are encountered, and are passed to the linkage editor whenever SMP/E needs to relink this load module.

SMP/E then scans the link-edit JCL for the link-edit attributes used to link this load module.

These are the link-edit attributes SMP/E recognizes in the PARM field and saves for future processing:

AC=1	AMODE=MIN	NCAL	RMODE=24
ALIGN2	AMOD=MIN	NE	REUS
AMODE=24	CALL	NOCALL	RMOD=24
AMOD=24	CASE(MIXED)	OL	RMODE=ANY
AMODE=31	CASE(UPPER)	OPTIONS( <i>ddname</i> )	RMOD=ANY
AMOD=31	DC	OVLY	REFR
AMODE=ANY	FETCHOPT(PACKINOPACK,PRIMEINOPRIME)		
AMOD=ANY	MAXBLK( <i>nnnnn</i> )	RENT	SCTR

When none of the above attributes are found, the STD indicator is set in the LMOD entry to indicate that the load module should be link-edited without any particular attributes.

**Notes:**

1. The OPTIONS attribute is recognized and processed, but it is not saved as part of the LMOD entry or the MOD entry being processed. It is used as a pointer to an imbedded file containing additional option specifications, allowing the PARM string to exceed the 100-character limit.
2. For more information on the attributes listed above, see “LMOD Entry (Distribution and Target Zone)” on page 698.

3. For more information on which attributes you can use with a specific link-edit utility, see the reference manual for that utility.

The LMOD entry constructed in this way contains the load module name, the libraries it resides in, the link-edit attributes, and the link-edit control statements required to relink the load module.

If SMP/E is creating an LMOD entry and finds an existing LMOD entry for the same load module containing only cross-zone subentries (a *stub* entry):

- SMP/E issues messages indicating that the cross-zone relationship might no longer be valid, and then deletes the cross-zone subentries from the LMOD entry.
- If deleting these cross-zone entries eliminates a TIEDTO relationship with a cross-zone, SMP/E deletes the associated TIEDTO value from the TARGETZONE entry for the set-to zone.
- Entries for related cross-zone modules are **not** updated to indicate that they are no longer part of the load module in the set-to zone, and the cross-zone's TIEDTO values are not updated.

## Processing Update Steps

Update steps are identified by one of the following:

- EXEC PGM=IEBUPDTE
- EXEC PGM=*updpgm* [if the JCLIN specified UPDATE(PGM=*updpgm*)]
- EXEC *updproc* [if the JCLIN specified UPDATE(*updproc*)]

When SMP/E recognizes an UPDATE step, it skips all the JCL until the SYSIN DD statement or the next EXEC statement is encountered. If the SYSIN statement is found, SMP/E skips all further input until one of the following is found:

- /\* or // is found if **SYSIN DD \*** was specified.
- /\* is found if **SYSIN DD DATA** was specified.
- *xx* is found if **SYSIN DD DATA,DLM=*xx*** was specified, where *xx* can be any two characters.

This is done so that, if the UPDATE step is adding JCL to a library, that JCL is not scanned as part of the JCLIN input.

## Processing Other Utility Steps

When SMP/E encounters an EXEC statement that does not specify one of the programs or cataloged procedures it recognizes, it skips all further JCL statements until the next EXEC statement is found. Input for certain system utility programs is not meant to be processed as JCLIN. However, the utility must be defined to SMP/E if it falls into any of these categories:

- Assembler
- Link-edit utility
- Copy
- Update

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of JCLIN processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.

2. JCLIN processing

Target zone	—	Update with exclusive enqueue.
-------------	---	--------------------------------

3. Termination

All resources are freed.



---

## Chapter 9. The LINK Command

Products sometimes contain modules from other products. For example, a product may need to:

- Include another product's modules in its load modules.

In this case, as long as the two products are in the same zone, SMP/E can automatically include the required modules in the load modules that need them (if the modules reside in the target library as single-CSECT load modules). SMP/E also tracks the inclusion of these cross-product modules in the load modules.

- Update another product's load module with one of its modules.

In this case, as long as the two products are in the same zone, SMP/E can automatically relink the load module and include the supplied module. SMP/E also tracks the inclusion of the modules in the cross-product load module.

When such products reside in different zones, however, SMP/E cannot automatically perform the cross-zone link-edits. Instead, you can use the LINK command to perform these cross-zone link-edits as postinstallation steps within SMP/E control. The LINK command causes the required load modules in one zone to be linked with modules residing in another zone, and tracks this inclusion so subsequent APPLY and RESTORE processing can automatically maintain the affected load modules.

### Notes:

1. The zones used by the LINK command must be defined in the same global zone.
2. Any zone updated with the LINK command or cross-zone information **cannot** be processed by SMP/E Release 6 or earlier.
3. When SMP/E processes the LINK command, it assumes that adding the desired modules to the load modules does not require any changes to the load module definition (that is, the link-edit utility control statements or link-edit utility attributes). If any such changes are needed, make them through JCLIN before using the LINK command.
4. There are times when the LINK command is **not** appropriate to use—generally, for products written in a high-level language and, as a result, include modules from libraries (such as compiler libraries) owned by a different product. Your options for installing such a product depend on how the product was packaged.

- SYSLIB DD statements are used in link-edit steps to implicitly include the necessary modules.

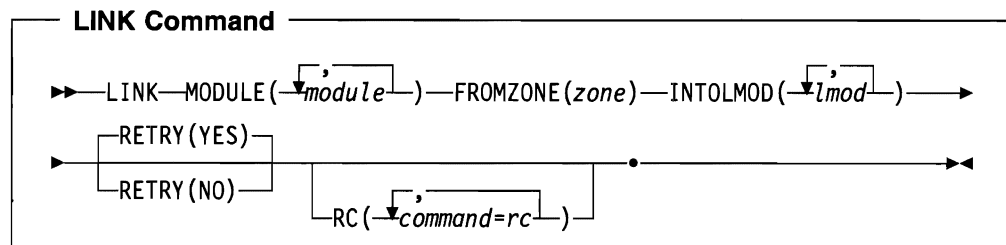
In this case, when you install the product, the implicitly-included modules are automatically linked into the load modules. If the libraries containing those modules are updated, you can use the REPORT CALLLIBS command to rebuild the affected load modules. For more information, see the Chapter 14, "The REPORT CALLLIBS Command" on page 275.

- No SYSLIB DD statements are used in link-edit steps in order to implicitly include the necessary modules. In this case, you must use postinstallation link-edit steps outside of SMP/E.

## Zones for SET BOUNDARY

For the LINK command, the SET BOUNDARY command must specify the target zone containing the LMOD entries for the load modules to be link-edited.

## Syntax



## Operands

### FROMZONE

specifies the zone containing the modules specified on the MODULE operand. The specified zone must be a target zone and must **not** be the same as the zone specified on the preceding SET command.

#### Notes:

1. FROMZONE is a required operand for the LINK command.
2. The zone specified on FROMZONE must be defined in the same global zone as the zone specified on the preceding SET command.

### INTOLMOD

specifies the load modules that should be link-edited to include the modules specified on the MODULE operand. These load modules must be defined in the zone specified on the preceding SET command.

#### Notes:

1. INTOLMOD is a required operand for the LINK command.
2. Do not specify a copied (single-CSECT) load module. You cannot use the LINK command to add modules to a single-CSECT load module; such load modules do not have any link-edit utility control statements that allow for the proper management of a multiple-module load module.

### MODULE

specifies the modules that should be linked into the load modules specified on the INTOLMOD operand.

#### Notes:

1. MODULE is a required operand for the LINK command.
2. Do not specify a module having the same name as a module installed in the set-to zone that is already part of the load module being updated.
3. You can specify a module that is from a totally copied library or that is a single-CSECT load module. However, you cannot specify a module that needs to be assembled.

**RC**

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the LINK command.

Before SMP/E processes the LINK command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the LINK command. Otherwise, the LINK command fails. For more information about the RC operand, see Appendix D.

**Notes:**

1. The RC operand **must be the last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the LINK command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

**RETRY**

indicates whether SMP/E should try to recover from out-of-space errors for utilities it calls.

**YES**

indicates that SMP/E should try to recover and retry the utility if a RETRYDDN list is available in the OPTIONS entry that is in effect. RETRY(YES) is the default.

If the retry attempt fails, the resulting x37 abend is treated as an unacceptable utility return code. In this case, processing continues for SYSMODs containing eligible updates to other libraries, but processing fails for SYSMODs containing unprocessed elements for the out-of-space library (and it fails for any SYSMODs that are dependent on the failed SYSMODs). For guidance on setting up the desired retry processing, see the *SMP/E R8.1 User's Guide*. For more information about OPTIONS entries, see "OPTIONS Entry (Global Zone)" on page 741.

If there is no RETRYDDN list, SMP/E does not try to recover from out-of-space errors, even if YES is specified.

**NO**

indicates that SMP/E should not try to recover from the error.

**Data Sets Used**

The following data sets may be needed to run the LINK command. They can be defined by DD statements or, preferably, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLTS	SYSRPT	SYSUT4
SMP_CSI	SMPOUT	SYSUT1	Distribution library
SMPLOG	SMPRPT	SYSUT2	Target library
SMPLOGA	SMPSNAP	SYSUT3	Zone

**Notes:**

1. The SMPLTS data set is required only if a load module having a CALLLIBS subentry list is being created, updated, deleted, or renamed.
2. *Distribution library* represents the DD statements required for each distribution library required to provide modules for the link-edits.
3. *Target library* represents the DD statements required for each target library required to provide modules or load modules for the link-edits.
4. *Zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated using the ZONEINDEX information in the GLOBALZONE entry.

**Output**

The File Allocation report is produced during LINK processing. This report is described in Chapter 31.

**Example: Linking a GDDM Module into a CICS Load Module**

Assume you have installed GDDM and CICS, and some of the GDDM\* modules must be linked into CICS\* load modules. GDDM resides in zone GDDTZN, and the zone controlling CICS is CICTZN. Because GDDM and CICS are controlled by different zones, SMP/E does not automatically link the GDDM modules into the CICS load modules when GDDM is installed. The LINK command can be used instead.

In this example, GDDM module ADMABCD needs to be linked into CICS load module DFHWXYZ. Module ADMABCD is installed in a target library as a single-CSECT load module when GDDM is installed. Therefore, SMP/E can use the target library version of ADMABCD to update CICS load module DFHWXYZ. (If a module does not reside in a target library as a single-CSECT load module, SMP/E uses the related distribution zone copy of the module to update the load module.)

The following commands can be used to have SMP/E install and track the installation of GDDM module ADMABCD in the CICS load module:

```
SET   BDY(CICTZN)           /* Target zone for CICS.   */.
LINK  MODULE(ADMABCD)      /* Link module ADMABCD    */
      FROMZONE(GDDTZN)    /* residing in zone GDDTZN */
      INTOLMOD(DFHWXYZ)   /* into load module DFHWXYZ. */.
```

These commands cause GDDM module ADMABCD to be linked into CICS load module DFHWXYZ. SMP/E also adds cross-zone subentries to the affected entries:

- An XZLMOD subentry is added to the ADMABCD MOD entry in target zone GDDTZN so that if ADMABCD is updated, it can be automatically replaced in the CICS load module.

**Note:** The CICS load module is automatically updated **only** if the XZLINK subentry was previously set to AUTOMATIC in the TZONE entry for zone



CICTZN. Here is an example of the commands that can be used to do this:

```
SET  BDY(CICTZN)      /* Target zone for CICS.  */.
UCLIN.
ADD  TZONE(CICTZN)   /* Update TZONE entry    */.
      XZLINK(AUTOMATIC). /* to do automatic links. */.
ENDUCL.
```

- An XZMOD subentry is added to the CICS DFHWXYZ LMOD entry in target zone CICTZN to indicate that:
  - DFHWXYZ now contains ADMABCD.
  - Any updates for ADMABCD should be accepted **only** from zone GDDTZN.
- TIEDTO subentries are added to the TZONE entries for CICTZN and GDDTZN to indicate that there is a relationship between modules and load modules in these zones.

---

## Processing

To process the LINK command, SMP/E first ensures that the syntax used for the LINK command is valid. Next, SMP/E checks whether both the zone specified on the FROMZONE operand and the zone specified on the preceding SET command are target zones. If they are, SMP/E obtains the CSIs containing the zones for update processing with an exclusive enqueue. Once SMP/E has obtained access to the CSI data sets, it opens the zones for update mode.

If SMP/E encounters any of the following errors, the LINK command fails:

- The LINK command contains syntax errors.
- Either the zone specified on the FROMZONE operand or the zone specified on the SET command (or both) is not a target zone.
- Errors were encountered while SMP/E was acquiring the CSIs or opening the zones.

Otherwise, SMP/E prepares to link the load modules and invokes the link-edit utility to do so.

## Preparing for Linking

Before invoking the link-edit utility, SMP/E checks whether it has access to the necessary modules, load modules, and libraries. If SMP/E encounters any errors that could cause the LINK command to fail, LINK command processing stops. Otherwise, SMP/E goes on to link-edit the load modules.

### Obtaining the Required Modules

To find a usable copy of each module specified on the MODULE operand, SMP/E checks the MOD entries in the target zone specified on the FROMZONE operand. SMP/E determines whether the modules have been installed in a target library (and are available for linking) by checking whether each MOD entry has both an FMID and an RMID value. If so, SMP/E checks the target zone MOD entry to determine

whether there is a stand-alone version of the module that can be used for the link-edit. A stand-alone version exists in either of these cases:

- The module was copied into a target library from a distribution library. (It is in a copied library.)
- The module exists by itself in a load module in a target library. (It is a single-CSECT load module.)

If SMP/E finds a stand-alone copy of the module, it saves the name of its target library for subsequent use in link-edit processing.

If SMP/E cannot find a stand-alone copy of a module in a target library, it checks the related distribution zone to determine whether a distribution library contains a usable copy. If there is a MOD entry in the distribution zone, SMP/E checks whether it contains both an FMID and an RMID value. If so, the module has been accepted into a distribution library, and a usable copy for the link-edit exists. SMP/E saves the name of the distribution library for subsequent use in link-editing the load module. In addition, SMP/E compares the distribution zone MOD entry to the related target zone MOD entry to determine whether they are at the same level. If they are at different levels, a warning message is issued, but the distribution zone copy of the module is still used for the link-edit.

### Notes:

1. If the two copies of the module are at different levels, you may want to synchronize the target zone version of the module with the distribution zone version. You can do this by accepting any applied SYSMODs that affect the module, or by restoring applied SYSMODs. Once the synchronization is done, you may want to use the LINK command again to relink the module into the load module.
2. SMP/E does not assemble a module in order to use it with the LINK command. The module must exist as a load module so it can serve as input to the link-edit utility.
3. If the load module already contains a copy of the module (for example, as a result of previous LINK processing), SMP/E does not check whether the copy of the module about to be linked into the load module is at an equal or higher level than the previous copy.

The following types of errors cause the LINK command to fail:

- No MOD entry was found in the FROMZONE target zone for a module specified on the MODULE operand.
- A module specified on the MODULE operand was not installed into a target library (its MOD entry in the FROMZONE target zone is missing either an FMID, an RMID, or both).
- There is no stand-alone version of a module in the target library, and one of the following errors was also found:
  - No related distribution zone is defined for the FROMZONE target zone.
  - SMP/E cannot obtain access to the related distribution zone.
  - No MOD entry exists in the distribution zone.

## Obtaining the Required Load Modules

SMP/E checks the LMOD entries for the load modules specified on the INTOLMOD operand to determine whether they can be processed by the LINK command. First, SMP/E verifies that none of the load modules are copied (single-CSECT) load modules. SMP/E then checks whether the LMOD entries already contain XZMOD subentries for any of the modules specified on the MODULE operand. These subentries indicate that the load module contains a module supplied by another zone. If SMP/E finds any such XZMOD entries, it verifies that the zone specified as the original supplier of the module is the same as the current FROMZONE value.

Next, SMP/E ensures that none of the modules specified on the MODULE operand will overlay a module by the same name in the set-to zone that is already part of a load module being updated and that is installed in the target libraries. For each LMOD entry found, SMP/E saves its target library information for subsequent use in the link operation.

The following types of errors cause the LINK command to fail:

- An LMOD entry does not exist.
- An LMOD entry exists only as XZMOD subentries.
- The load module has been copied (it is a single-CSECT load module).
- An LMOD entry contains an XZMOD subentry for a module specified on the MODULE operand, but the zone specified as the original supplier of the module is different from the current FROMZONE value.
- A module specified on the MODULE operand has the same name as a module from the set-to zone that is already part of a load module being updated and that is installed in the target libraries.

## Checking the Libraries for the Modules and Load Modules

SMP/E makes sure it can allocate the required libraries (the target libraries for the load modules to be updated and the target or distribution libraries containing the modules to be included). If a DD statement was not specified for a library, SMP/E attempts to allocate it dynamically using the appropriate DDDEF entry, as follows:

- For load module libraries, SMP/E uses the DDDEF entries in the zone specified on the preceding SET command. If an LMOD entry contains a CALLLIBS subentry list, the zone containing the LMOD entry must also contain a DDDEF entry for each CALLLIBS library.
- For module libraries, SMP/E uses the following DDDEF entries:
  - For target libraries, it uses DDDEF entries in the zone specified on the FROMZONE operand.
  - For any necessary distribution libraries, it uses DDDEF entries in the DLIB zone related to the FROMZONE.

Next, SMP/E verifies that all the load modules to be link-edited are actually in the indicated target libraries. If a load module is not in its library, it is not link-edited from that library. If it is not in any of the indicated target libraries, it is not link-edited at all. A load module with a CALLLIBS subentry must exist in the SMPLTS data set, although it is not required that it already exist in the target libraries.

The following types of errors cause the LINK command to fail:

- A required DD statement is missing and the associated data set cannot be dynamically allocated.
- None of the load modules to be updated are in the indicated target libraries.

## Linking the Load Modules

SMP/E invokes the link-edit utility for each load module to be updated.

### Notes:

1. SMP/E recognizes IEANUC01 as a special load module, so it performs special processing whenever IEANUC01 is to be updated. When SMP/E determines that IEANUC01 is to be relinked, it saves a backup copy (IEANUC0x), where x is the NUCID value in the current OPTIONS entry. In certain cases, however, a backup copy is not made or may be lost:
  - If the OPTIONS entry has no NUCID value or if the value is 1, **no** backup copy of the nucleus is created.
  - If two LINK commands are processed that each cause the nucleus to be replaced, and if the NUCID value is not changed between the first and second LINK, the backup copy from the first LINK is lost.
  - If a combination of APPLY and LINK commands are processed so that each causes the nucleus to be replaced, and if the NUCID value is not changed between the two commands, the backup copy from the first command is lost.
2. Make sure your nucleus data set is large enough to contain as many copies of the nucleus as required (minimum of three). The backup copy of the nucleus is not used by SMP/E to restore a SYSMOD. It is created so you can use it to IPL an alternative nucleus if you are not able to IPL the system after running the LINK command.

For each successful link-edit, SMP/E updates the MOD and LMOD entries involved:

- The MOD entries are updated with XZLMOD subentries to indicate the cross-zone load modules they were linked into. XZLMOD subentries specify the name of the load module and the zone name specified on the previous SET operand.
- The LMOD entries are updated with XZMOD subentries to indicate the cross-zone modules they now contain. XZMOD subentries specify the name of the module and the zone name specified on the FROMZONE operand.

SMP/E adds TIEDTO values to record the relationship between the zone specified on the FROMZONE operand and the zone specified on the preceding SET command:

- The zone name from the SET command becomes a TIEDTO subentry in the TARGETZONE entry for the zone specified on the FROMZONE operand.
- The zone name from the FROMZONE operand becomes a TIEDTO subentry in the TARGETZONE entry for the zone specified on the preceding SET command.

These subentries define the cross-zone relationship between the modules and load modules and are used during APPLY and RESTORE processing to update the load modules when the modules are updated. For more information about these subentries, see “LMOD Entry (Distribution and Target Zone)” on page 698 and “MOD Entry (Distribution and Target Zone)” on page 723.

Utility failures can cause the LINK command to fail. For details on handling x37 abends, see the description of the RETRY operand under “Operands” on page 202.

---

## Zone and Data Set Sharing Considerations

This section shows the phases of LINK processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

- Global zone – Read without enqueue.
- Target zones – Read without enqueue.

### 2. LINK processing

- Global zone – Read without enqueue.
- Target zones – Update with exclusive enqueue.
- DLIB zone (as required) – Read with shared enqueue.

### 3. Termination

All resources are freed.



---

## Chapter 10. The LIST Command

The SMP/E data sets contain a great deal of information—the global zone, target zones, distribution zones, SMPPTS, SMPLOG, and SMPSCDS—that you may find useful when installing a new function, preparing a user modification, or debugging a problem. You can use the SMP/E LIST command to display that information.

SMP/E can display all the entries of a specified type (such as MOD, MAC, SYSMOD, and so on), or it can display information for selected entries. In addition, for SYSMOD entries, SMP/E provides some additional operands you can specify to list groups of SYSMODs that meet certain criteria.

---

### Zones for SET BOUNDARY

To list entries in a CSI data set, you must specify the name of the zone containing the entries to be listed on the SET BOUNDARY command.

To list entries in a data set other than the CSI (such as the SMPLOG or SMPSCDS), you must specify the zone associated with that data set on the SET BOUNDARY command:

- **SMPLOG:** Specify the zone containing the DDDEF entry for the particular SMPLOG data set to be listed.
- **SMPSCDS:** Specify the target zone containing the DDDEF entry for the particular SMPSCDS data set to be listed.

Make sure the data you request to have listed is valid for the specified zone type.

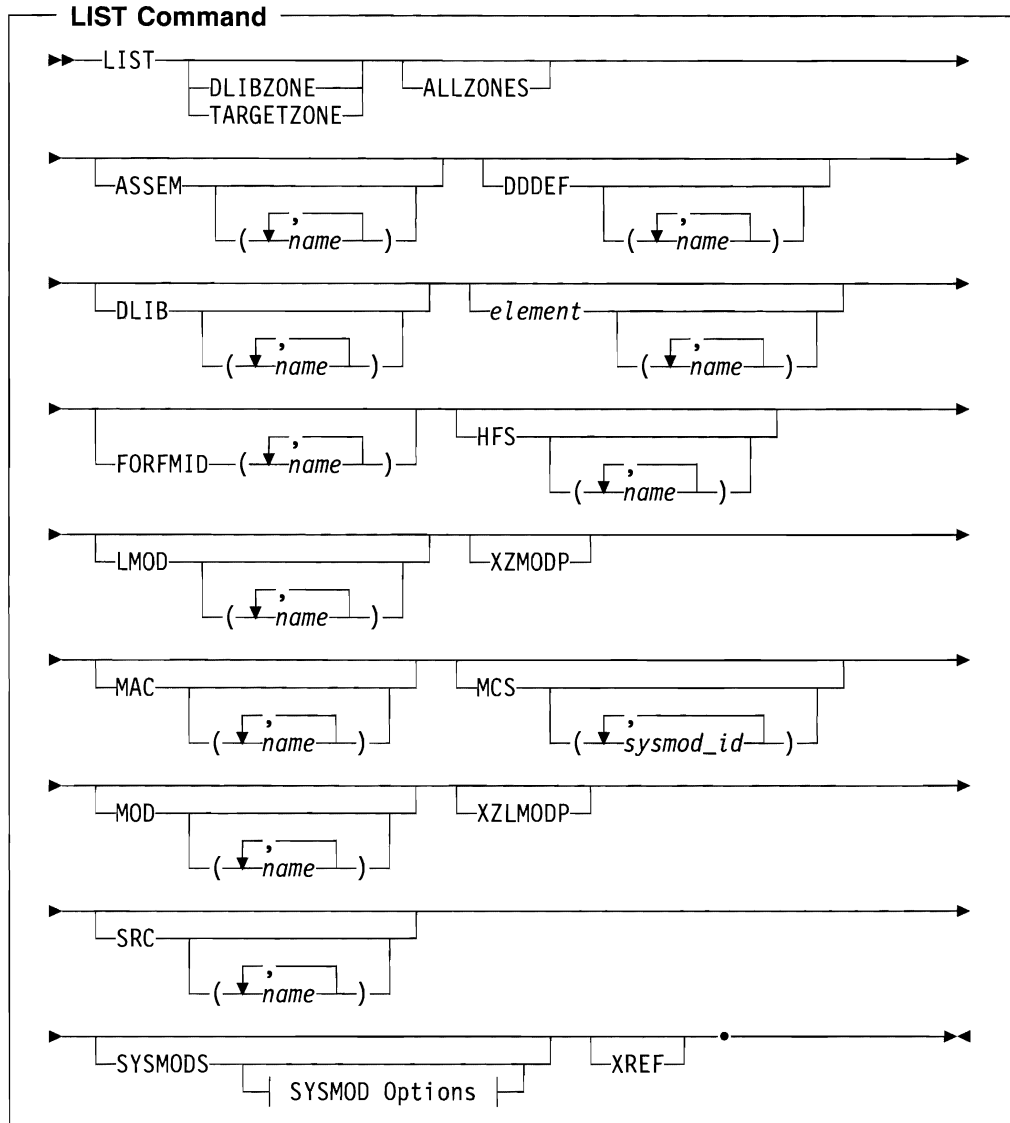
---

### Syntax

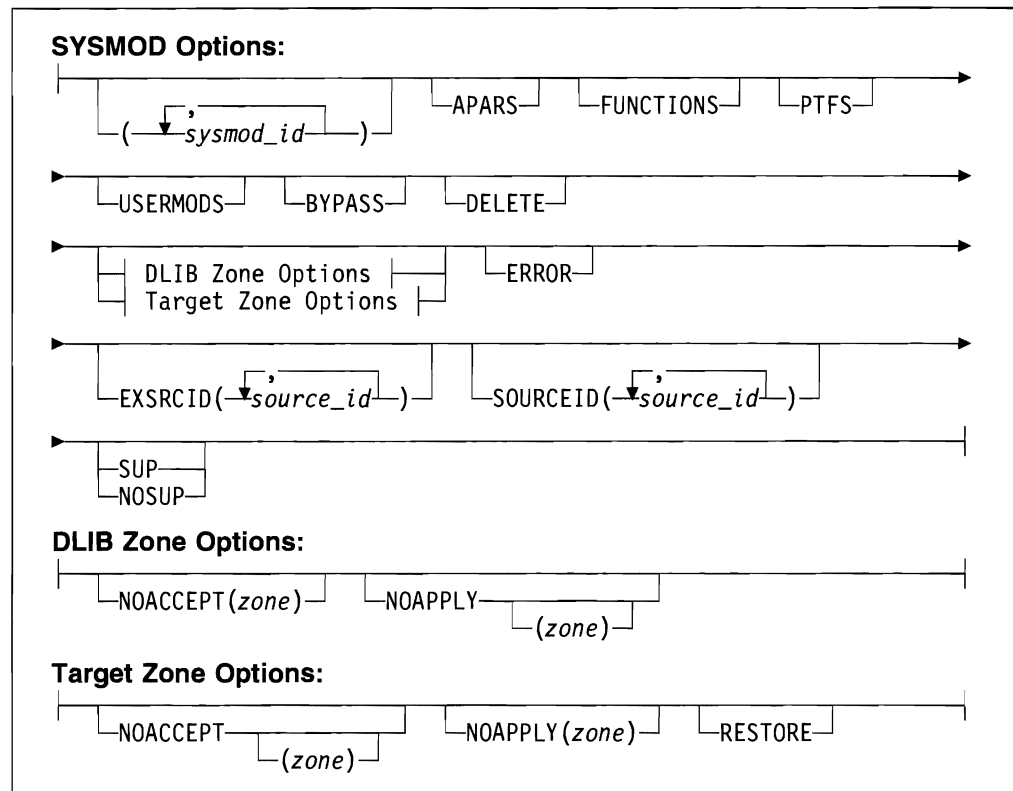
This section shows the LIST command syntax for the following zones and data sets:

- Distribution and target zones
- Global zone
- SMPLOG
- SMPSCDS

## Distribution Zone and Target Zone Syntax





**Notes:**

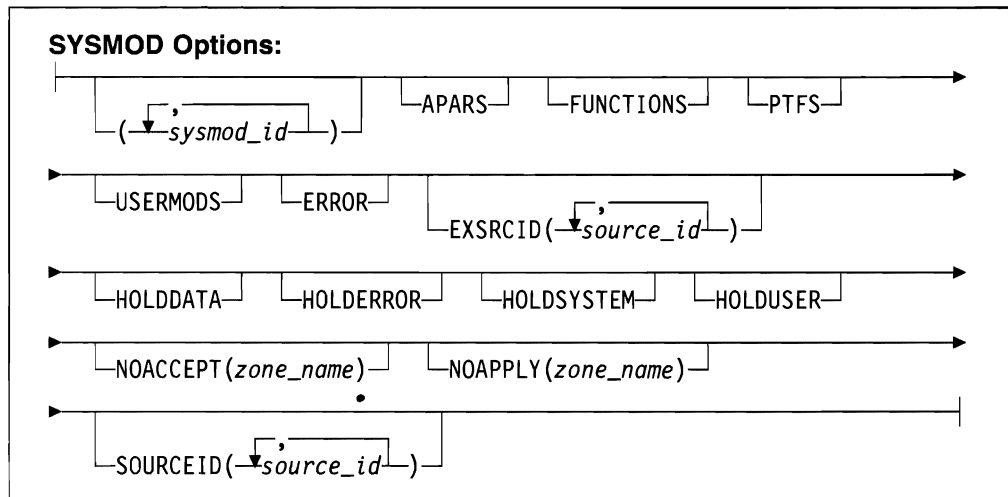
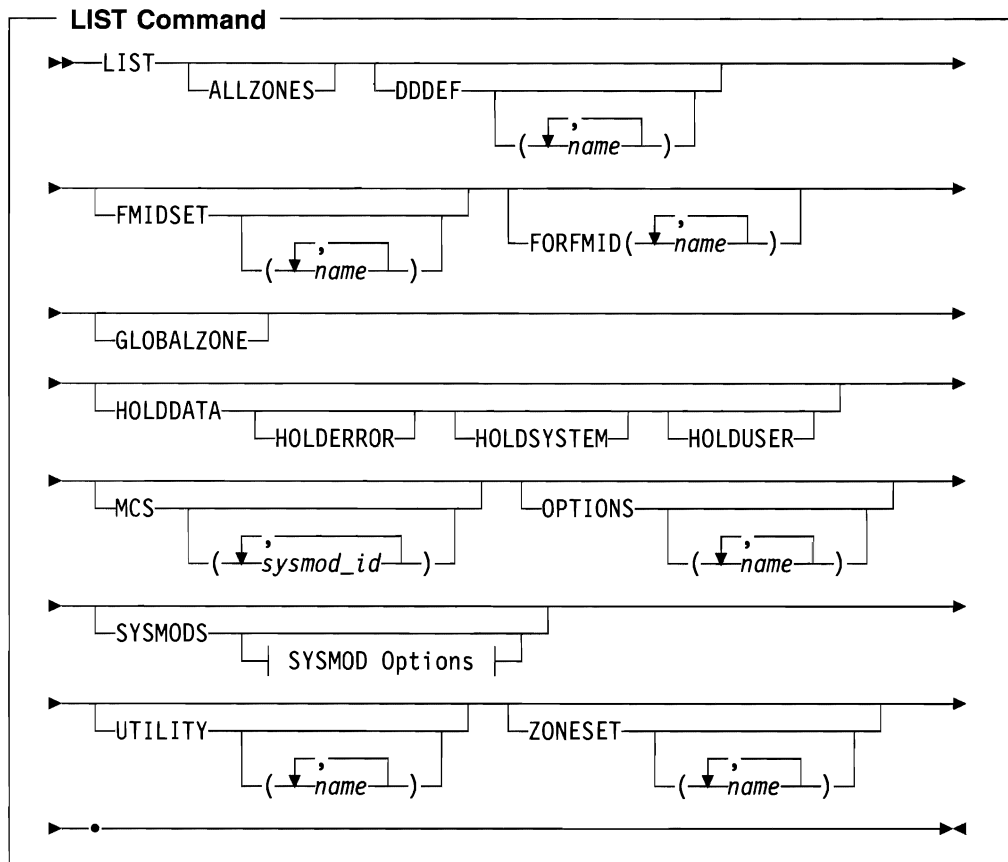
1. The SYSMODS operand is optional if you specify any of the following operands:

APARS	FUNCTIONS	PTFS
BYPASS	MCS	RESTORE
DELETE	NOACCEPT	SOURCEID
ERROR	NOAPPLY	SUP
EXSRCID	NOSUP	USERMODS

2. The XZLMODP and XZMODP operands are valid only for target zone entries.

See the descriptions of the operands for details on all the operands.

## Global Zone Syntax

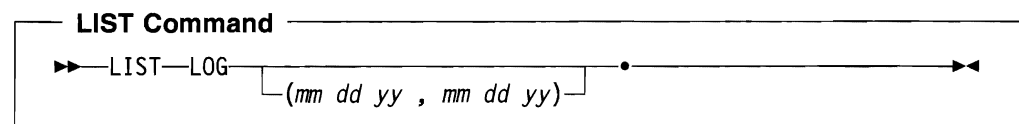


**Note:** The SYSMODS operand is optional if you specify any of the following operands:

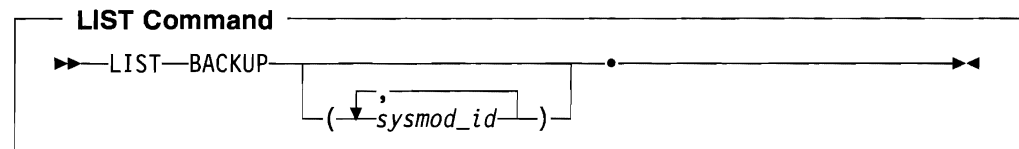
APARS	HOLDDATA	NOACCEPT
ERROR	HOLDERROR	NOAPPLY
EXSRCID	HOLDSYSTEM	PTFS
FUNCTIONS	HOLDUSER	SOURCEID
		USERMODS

See the operand descriptions for details on all the operands.

## SMPLOG Syntax



## SMPSCDS Syntax



## Operands

### ALLZONES

indicates that SMP/E should list information from the global zone and all the target and distribution zones defined by ZONEINDEX subentries.

#### Notes:

1. ALLZONES is mutually exclusive with NOAPPLY, NOACCEPT, HOLDDATA, HOLDERROR, HOLDSYSTEM, HOLDUSER, and MCS.
2. You can limit the information to be listed by specifying only the entries or entry types that you need. For example:

```
SET    BDY(GLOBAL)    /* set to global zone    */.
LIST   SYSMOD(UZ12345)/* list this SYSMOD entry */
        ALLZONES      /* from wherever it is  */.
```

lists SYSMOD entry UZ12345 in each zone to which it has been applied or accepted.

3. ALLZONES is allowed when the SET command specifies the global zone, a target zone, or a distribution zone.

The entries listed are the same, regardless of the type of zone you specify, because the output is determined by the additional operands on the LIST command and by the entry types valid within each zone to be listed. For example, the following lists module X in all target and distribution zones:

```
LIST ALLZONES MOD(X).
```

The global zone is skipped, because there are no modules in the global zone.

### APARS

indicates that SMP/E should list APAR SYSMODs.

#### Notes:

1. **APARS** can also be specified as **APAR**.
2. When **APARS** is used with **FUNCTIONS**, **PTFS**, or **USERMODS**, SMP/E lists any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** was also specified, even if it was not.

### ASSEM

indicates that SMP/E should list all ASSEM entries or the specified ASSEM entries.

**Note:** ASSEM is allowed when the SET command specifies a target zone or distribution zone.

### BACKUP

indicates that SMP/E should list all BACKUP entries or the specified BACKUP entries.

#### Notes:

1. BACKUP is mutually exclusive with all other LIST operands.
2. BACKUP is allowed when the SET command specifies a target zone.

### BYPASS

indicates that SMP/E should list entries for SYSMODs installed using the BYPASS operand.

#### Notes:

1. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** was also specified, even if it was not.
2. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

### DDDEF

indicates that SMP/E should list all DDDEF entries or the specified DDDEF entries.

### DELETE

indicates that SMP/E should list entries for function SYSMODs that have been explicitly deleted from the target zone or distribution zone by other function SYSMODs.

#### Notes:

1. DELETE is allowed when the SET command specifies a target zone or distribution zone.
2. **DELETE** can also be specified as **DEL**.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** was also specified, even if it was not.

**DLIB**

indicates that SMP/E should list all DLIB entries or the specified DLIB entries.

**Note:** DLIB is allowed when the SET command specifies a target zone or distribution zone.

**DLIBZONE**

indicates that SMP/E should list the DLIBZONE entry.

**Notes:**

1. DLIBZONE is allowed when the SET command specifies a distribution zone.
2. **DLIBZONE** can also be specified as **DZONE**.

*element*

is used to list a particular type of data element entry. *element* indicates that SMP/E should list all data element entries of that type or the specified data element entries.

**Notes:**

1. *element* is allowed when the SET command specifies a target zone or distribution zone.
2. Table 26 on page 523 shows the types of data elements that can be specified for the *element* operand.
3. Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *element* operand does not contain any *xxx* value.) Table 27 on page 526 shows the *xxx* values and the languages they represent.

**ERROR**

indicates that SMP/E should list SYSMOD entries in which the ERROR indicator is set.

**Notes:**

1. ERROR is allowed when the SET command specifies a target zone or distribution zone.
2. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** was also specified, even if it was not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

**EXSRCID**

indicates that SYSMODs associated with the specified source IDs should **not** be listed.

### Notes:

1. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** was also specified, even if it was not.
2. There are two ways to specify source IDs:
  - Explicitly, by specifying a particular source ID—for example, **PUT9303**. In this case, only that particular source ID is used.
  - Implicitly, by specifying either **\*** or **c\*** (for example, **PUT\***), where *c* is a 1- to 7-character string. In the second case, all source IDs beginning with the specified character string are used.
3. A given source ID can be explicitly specified **only once** on the EXSRCID operand.
4. The same source ID **cannot** be explicitly specified on both the EXSRCID and SOURCEID operands.
5. If a source ID is specified, implicitly or explicitly, on the EXSRCID operand and on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.
6. If a given SYSMOD has multiple source IDs and at least one of those source IDs is specified implicitly or explicitly on the source ID operand, the SYSMOD is excluded from processing if another one of its source IDs is specified implicitly or explicitly on the EXSRCID operand.  
  
For example, assume PTF UZ12345 has been assigned source IDs SMCREC and PUT9303. If you specify **SOURCEID(SMC\*)** and **EXSRCID(PUT9303)**, the SYSMOD is excluded from processing.
7. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

### FMIDSET

indicates that SMP/E should list all FMIDSET entries or the specified FMIDSET entries.

### Notes:

1. FMIDSET is allowed when the SET command specifies the global zone.
2. **FMIDSET** can also be specified as **FMSET**.
3. To list element and SYSMOD entries owned by an FMID defined in a particular FMIDSET entry, use the FORFMID operand, not FMIDSET. The FMIDSET operand provides a listing only of the specified FMIDSET entries, not a listing of the entries owned by FMIDs defined in the specified FMIDSET entries.

### FORFMID

indicates that SMP/E should list only entries currently owned by one of the specified FMIDs or by an FMID defined in one of the specified FMIDSET entries.

**Notes:**

1. You can specify FMIDs, FMIDSET entries, or both.
2. Only element and SYSMOD entries are listed by the FORFMID operand.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** was also specified, even if it was not, **unless** an element type operand was also specified. In that case, FORFMID limits the element entries that are listed.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.
5. FORFMID with the HOLDDATA operand lists only SYSMODs with the specified FMID that have been received.

**FUNCTIONS**

indicates that SMP/E should list function SYSMODs.

**Notes:**

1. **FUNCTIONS** can also be specified as **FUNCTION**.
2. When **FUNCTIONS** is used with **APARS**, **PTFS**, or **USERMODS**, SMP/E lists any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** was also specified, even if it was not.

**GLOBALZONE**

indicates that SMP/E should list the GLOBALZONE entry.

**Notes:**

1. GLOBALZONE is allowed when the SET command specifies the global zone.
2. **GLOBALZONE** can also be specified as **GZONE**.

**HFS**

indicates that SMP/E should list all HFS entries or the specified HFS entries.

**HOLDDATA**

When specified with the SYSMOD operand, HOLDDATA indicates that SMP/E should list the ++HOLD MCSs associated with the SYSMOD entries that are listed.

When specified without the SYSMOD operand, HOLDDATA indicates that SMP/E should list all HOLDDATA entries. No SYSMOD entries are listed.

You can limit which HOLDDATA entries are listed by coding one or more of the following operands:

HOLDERROR  
HOLDSYSTEM  
HOLDUSER

If you specify more than one type of hold, SMP/E lists only entries containing holds for **all** the specified types. For example, the following commands list all HOLDDATA entries with **both** HOLDERROR and HOLDSYSTEM reason IDs:

```

SET      BDY(GLOBAL) /* set to global zone */.
LIST     HOLDDATA    /* list only the HOLDDATA */
         HOLDERROR   /* entries that contain */
         HOLDSYSTEM  /* both error and system */
         /* holds    */ /*./
    
```

**Notes:**

1. HOLDDATA is allowed when the SET command specifies the global zone.
2. HOLDDATA with the FORFMID operand lists only SYSMODs with the specified FMID that have been received.
3. Table 15 summarizes the LIST results for various combinations of the HOLDDATA operand with other related operands.

<i>Table 15. Information Listed for HOLDDATA Combined with Other Operands</i>		
<b>HOLD-Related Operands</b>	<b>Information Listed When the SYSMOD Operand Is Specified</b>	<b>Information Listed When the SYSMOD Operand Is Not Specified</b>
HOLDDATA  (without HOLDERROR, HOLDSYSTEM, or HOLDUSER)	SYSMOD entries plus all associated ++HOLD statements  No HOLDDATA entries	HOLDDATA entries  No SYSMOD entries
HOLDDATA  (with HOLDERROR, HOLDSYSTEM, or HOLDUSER)	SYSMOD entries plus all associated ++HOLD statements  (++HOLD statements not limited to specified type of reason ID)  No HOLDDATA entries	HOLDDATA entries that contain <b>all</b> the specified types of hold reason IDs
HOLDERROR, HOLDSYSTEM, or HOLDUSER  (without HOLDDATA)	No ++HOLD statements included with the SYSMOD entries  No HOLDDATA entries	SYSMOD entries associated with the specified type of reason ID  No ++HOLD statements included with the SYSMOD entries  No HOLDDATA entries

**HOLDERROR**

When specified without the HOLDDATA operand (and either with or without the SYSMOD operand), HOLDERROR indicates that SMP/E should list only SYSMODs associated with error hold reason IDs. The associated ++HOLD MCSs are not listed.

**Note:** If the reason IDs are bypassed or resolved, these SYSMODs might not actually be held during APPLY or ACCEPT processing.

When specified with the HOLDDATA operand but without the SYSMOD operand, HOLDERROR indicates that HOLDDATA entries for error hold reason IDs should be listed. No SYSMOD entries are listed.



**Notes:**

1. HOLDERROR is allowed when the SET command specifies the global zone.
2. HOLDERROR can also be specified as HOLDERR.

**HOLDSYSTEM**

When specified without the HOLDDATA operand (and either with or without the SYSMOD operand), HOLDSYSTEM indicates that SMP/E should list only SYSMODs associated with system hold reason IDs. The associated ++HOLD MCSs are not listed.

**Note:** If the reason IDs are bypassed or resolved, these SYSMODs might not actually be held during APPLY or ACCEPT processing.

When specified with the HOLDDATA operand but without the SYSMOD operand, HOLDSYSTEM indicates that SMP/E should list HOLDDATA entries for system hold reason IDs. No SYSMOD entries are listed.

**Notes:**

1. HOLDSYSTEM is allowed when the SET command specifies the global zone.
2. HOLDSYSTEM can also be specified as HOLDSYS.

**HOLDUSER**

When specified without the HOLDDATA operand (and either with or without the SYSMOD operand), HOLDUSER indicates that SMP/E should list only SYSMODs associated with user hold reason IDs. The associated ++HOLD MCSs are not listed.

**Note:** If the reason IDs are bypassed or resolved, these SYSMODs might not actually be held during APPLY or ACCEPT processing.

When specified with the HOLDDATA operand but without the SYSMOD operand, HOLDUSER indicates that HOLDDATA entries for user hold reason IDs should be listed. No SYSMOD entries are listed.

**Note:** HOLDUSER is allowed when the SET command specifies the global zone.

**LMOD**

indicates that SMP/E should list all LMOD entries or the specified LMOD entries.

**Note:** LMOD is allowed when the SET command specifies a target zone or distribution zone.

**LOG**

indicates that SMP/E should list either the total contents of the LOG or the contents within a selected date range.

*(mm dd yy, mm dd yy)* specifies a range of dates within the data set to be listed. If no date range is specified, the contents of the entire LOG data set are listed.

The dates are specified as *mm dd yy*, where *mm* is the month (01–12), *dd* is the day (01–31), and *yy* is the year (00–99). Blanks separate the month, day, and year.

The following commands list the data in the LOG for June 8 through June 11, 1994:

```
SET      BDY(TGT1)      /* set to target zone    */.
LIST     LOG(06 08 94  /* list log within this  */.
         06 11 94) /* date range            */.
```

These commands list the data in the LOG for one day, June 9, 1994:

```
SET      BDY(TGT1)      /* set to target zone    */.
LIST     LOG(06 09 94  /* list log for this one */.
         06 09 94) /* day                   */.
```

#### Notes:

1. LOG is mutually exclusive with all other LIST operands.
2. To determine which LOG data set to list, SMP/E checks the SMPLOG DDDEF entry in the zone specified on the SET command.
3. SMP/E views its LOG data set as one "logical" data set, even though there might actually be two separate physical data sets: SMPLOG and SMPLOGA. So, if an SMPLOGA DDDEF is defined in the zone and data has spilled over from the SMPLOG data set into the SMPLOGA data set, LIST LOG also lists the contents of the SMPLOGA data set. You can also specify a date range that spans the SMPLOG and SMPLOGA data sets, or a date range that is only in the SMPLOGA data set, because SMP/E views the two data sets as a single "logical" data set.

#### MAC

indicates that SMP/E should list all MAC entries or the specified MAC entries.

#### MCS

specifies that SMP/E should list all MCS entries or the specified MCS entries. LIST MCS can be used to print PTF cover letters.

- If no SYSMOD IDs are specified, SMP/E lists the MCSs associated with all the SYSMOD entries in the current zone.
- If SYSMOD IDs are specified, SMP/E lists only the MCSs for the specified SYSMOD entries. For example, the following commands list only the MCSs for AZ12345:

```
SET      BDY(TGT1)      /* set to target zone    */.
LIST     APAR           /* list all APAR type    */.
         SYSMOD         /* SYSMODs plus         */.
         MCS(AZ12345)  /* this one MCS entry    */.
```

#### MOD

indicates that SMP/E should list all MOD entries or the specified MOD entries.

#### NOACCEPT

indicates that SMP/E should list SYSMOD entries from the current zone that are **not** accepted into a particular distribution zone. You can use NOACCEPT to:

- See which SYSMODs have been received but have not yet been accepted into the specified distribution zone.

To do this, specify the global zone on the SET command and the distribution zone you want to check on the NOACCEPT operand.

- See which SYSMODs have been applied in a particular target zone but have not yet been accepted into one of the following:

- Its related distribution zone
- The distribution zone specified on NOACCEPT

To do this, specify the desired target zone on the SET command and do one of the following:

- To check for SYSMODs that have not been accepted into the related distribution zone, specify NOACCEPT without a zone name.
- To check for SYSMODs that have not been accepted into a particular distribution zone, specify NOACCEPT with the appropriate distribution zone name.

- Compare which SYSMODs are accepted in two distribution zones.

To do this, specify one distribution zone on the SET command and the other on the NOACCEPT operand. SMP/E lists the SYSMODs that have been accepted into the set-to zone, but not into the NOACCEPT zone.

For examples, see “Examples” on page 231.

#### Notes:

1. **NOACCEPT** can also be specified as **NOACC**.
2. If you specify either the global zone or a distribution zone on the SET command, you **must** specify a distribution zone on NOACCEPT.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.
5. You can also use the REPORT SYSMODS command to compare zones and to generate the commands needed to install SYSMODs in the zone where they are not installed. See Chapter 18 for more information.

#### NOAPPLY

indicates that SMP/E should list SYSMOD entries from the current zone that are **not** applied to a particular target zone. You can use NOAPPLY to:

- See which SYSMODs have been received but have not yet been applied to the specified target zone.

To do this, specify the global zone on the SET command and the target zone you want to check on the NOAPPLY operand.

- See which SYSMODs have been accepted into a particular distribution zone but have not yet been applied to one of the following:
  - Its related target zone
  - The target zone specified on NOAPPLY

To do this, specify the desired distribution zone on the SET command and do one of the following:

- To check for SYSMODs that have not been applied to the related target zone, specify NOAPPLY without a zone name.

- To check for SYSMODs that have not been applied to a particular target zone, specify NOAPPLY with the appropriate target zone name.
- Compare which SYSMODs are applied to two target zones.  
To do this, specify one target zone on the SET command and the other on the NOAPPLY operand. SMP/E lists the SYSMODs that have been applied to the set-to zone but not to the NOAPPLY zone.

For more information, see “Examples” on page 231.

### Notes:

1. **NOAPPLY** can also be specified as **NOAPP**.
2. If you specify either the global zone or a target zone on the SET command, you **must** specify a target zone on NOAPPLY.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** was also specified, even if it was not.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.
5. You can also use the REPORT SYSMODS command to compare zones and to generate the commands needed to install SYSMODs in the zone where they are not installed. See Chapter 18 for more information.

### NOSUP

indicates that SMP/E should list entries for SYSMODs that have not been superseded.

### Notes:

1. NOSUP is mutually exclusive with SUP.
2. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

### OPTIONS

indicates that SMP/E should list all OPTIONS entries or the specified OPTIONS entries.

**Note:** OPTIONS is allowed when the SET command specifies the global zone.

### PTFS

indicates that SMP/E should list PTF SYSMODs.

### Notes:

1. **PTFS** can also be specified as **PTF**.
2. When **PTFS** is used with **APARS**, **FUNCTIONS**, or **USERMODS**, SMP/E lists any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** has also been specified, even if it has not.

**RESTORE**

indicates that SMP/E should list SYSMOD entries in which the RESTORE indicator is set. These SYSMODs have been incompletely restored and are in error.

**Notes:**

1. RESTORE is allowed when the SET command specifies a target zone.
2. **RESTORE** can also be specified as **RES**.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

**SOURCEID**

indicates that SMP/E should list only SYSMOD entries associated with one of the specified SOURCEID values.

**Notes:**

1. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.
2. There are two ways to specify source IDs:
  - Explicitly, by specifying a particular SOURCEID (for example, **PUT9303**). In this case, only that particular SOURCEID is used.
  - Implicitly, by specifying either \* or c\* (for example, **PUT\***), where c is a 1- to 7-character string. In the second case, all source IDs beginning with the specified character string are used.
3. A given source ID can be explicitly specified **only once** on the source ID operand.
4. The same source ID **cannot** be explicitly specified on both the EXSRCID and SOURCEID operands.
5. If a source ID is specified, implicitly or explicitly, on the SOURCEID operand and also on the EXSRCID operand, all SYSMODs with that source ID are excluded from processing.
6. If a given SYSMOD has multiple source IDs of which at least one is specified either implicitly or explicitly on the SOURCEID operand and another is specified either implicitly or explicitly on the EXSRCID operand, the SYSMOD is excluded from processing.

For example, assume PTF UZ12345 has been assigned source IDs SMCREC and PUT9303. If you specify **SOURCEID(SMC\*)** and **EXSRCID(PUT9303)**, the SYSMOD is excluded from processing.

7. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

**SRC**

indicates that SMP/E should list all SRC entries or the specified SRC entries.

**SUP**

indicates that SMP/E should list entries for SYSMODs that have been superseded.

**Notes:**

1. SUP is mutually exclusive with NOSUP.
2. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** was also specified, even if it was not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.
4. To list "dummy" entries for superseded SYSMODs (entries for SYSMODs that were superseded but not installed), do **not** specify a SYSMOD type operand. No SYSMOD type is associated with such entries.

**SYSMODS**

indicates that SMP/E should list all SYSMOD entries or the specified SYSMOD entries.

You can limit which SYSMOD entries are listed by coding one or more of the following SYSMOD qualifier operands:

APARS, FUNCTIONS, PTFS, or USERMODS  
 BYPASS  
 DELETE  
 ERROR  
 EXSRCID  
 FORFMID  
 HOLDDATA  
 HOLDERERROR, HOLDSYSTEM, or HOLDUSER  
 MCS  
 NOACCEPT  
 NOAPPLY  
 NOSUP or SUP  
 RESTORE  
 SOURCEID

For the operands shown on separate lines, SMP/E lists only SYSMOD entries that meet **all** the specified criteria. For the operands shown on the same line, SMP/E lists SYSMOD entries that meet **any one** of the specified criteria. For example, the following commands list all APAR SYSMODs that were previously installed and subsequently have been superseded:

```
SET      BDY(TGT1)      /* set to target zone      */
LIST     SYSMOD        /* list SYSMODs            */
         APAR          /* that are APARs         */
         SUP           /* and are superseded     */
```

On the other hand, these commands list APAR or function SYSMODs that were previously installed and subsequently have been superseded:

```

SET      BDY(TGT1)      /* set to target zone    */.
LIST     SYSMOD        /* list SYSMODs          */.
         APAR          /* that are APARs       */.
         FUNCTION      /* or functions         */.
         SUP           /* and are superseded   */.

```

**Notes:**

1. **SYSMODS** can also be specified as **SYSMOD**.
2. If any of the SYSMOD qualifier operands (other than HOLDDATA or MCS) are specified without the SYSMOD operand, SMP/E assumes that you want the SYSMOD entries listed and, therefore, processes as if **SYSMOD** was also specified.
3. When either **MCS** or **HOLDDATA** is specified without a name list on the same LIST command as the SYSMOD operand, SMP/E assumes you want the MCS entries or HOLDDATA only for those SYSMOD entries that are listed, not all the MCS and HOLDDATA entries. If you want to list **all** the MCS or HOLDDATA entries, use the LIST command with the MCS or HOLDDATA operand, but without the SYSMOD operand and without any of the SYSMOD qualifier operands identified above.

**TARGETZONE**

indicates that SMP/E should list the TARGETZONE entry.

**Notes:**

1. TARGETZONE is allowed when the SET command specifies a target zone.
2. TARGETZONE can also be specified as TZONE.
3. The XZLINK(DEFERRED) value is displayed only when the TARGETZONE entry contains TIEDTO records.

**USERMODS**

indicates that SMP/E should list USERMOD SYSMODs.

**Notes:**

1. USERMODS can also be specified as USERMOD.
2. When USERMODS is used with **APARS**, **FUNCTIONS**, or **PTFS**, SMP/E lists any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.

**UTILITY**

indicates that SMP/E should list all UTILITY entries or the specified UTILITY entries.

**Note:** UTILITY is allowed when the SET command specifies the global zone.

**XREF**

generates cross-reference information appropriate to the entry type being listed. Table 16 shows the information included for each entry type:

Entry Type	XREF Information
ASSEM entries	A list of all macros whose MAC entry indicates that this module should be reassembled
Element entries	A history of all SYSMODs affecting this element
LMOD entries	A list of all MOD entries that are linked or copied to this load module
SYSMOD entries	<ul style="list-style-type: none"> <li>• A list of all SYSMODs specifying this SYSMOD on the ++VER DELETE operand</li> <li>• A list of all SYSMODs specifying this SYSMOD on the ++VER NPRES operand</li> <li>• A list of all SYSMODs specifying this SYSMOD on the ++VER PRE operand</li> <li>• A list of all SYSMODs specifying this SYSMOD on the ++VER REQ operand</li> <li>• A list of all SYSMODs specifying this SYSMOD on the ++VER SUP operand</li> <li>• A list of all SYSMODs specifying this SYSMOD on the ++VER VERSION operand</li> <li>• A list of all SYSMODs specifying this SYSMOD on the ++IF REQ operand</li> </ul>

**Note:** SMP/E uses extra time and more storage to generate the additional data requested by the XREF operand.

**XZLMODP**

indicates that SMP/E should list MOD entries for all modules that have been linked into load modules controlled by a different target zone. (The MOD entries for these modules contain XZLMODP subentries.)

**Notes:**

1. XZLMODP is allowed only when the SET command specifies a target zone.
2. The appropriate MOD entries are listed, regardless of whether the MOD operand was specified on the LIST command.
3. If both MOD and XZLMODP are specified, only MODs with cross-zone subentries are listed. If a list of MODs and XZLMODP is specified, all the specified MODs, as well as all the MODs with cross-zone subentries, are listed.

**XZMODP**

indicates that SMP/E should list LMOD entries for all load modules containing modules from a different target zone. (The LMOD entries for these load modules contain XZMODP subentries.)



**Notes:**

1. XZMODP is allowed only when the SET command specifies a target zone.
2. The appropriate LMOD entries are listed regardless of whether the LMOD operand was specified on the LIST command.
3. If both LMOD and XZMODP are specified, only LMODs with cross-zone subentries are listed. If a list of LMODs and XZMODP is specified, all the specified LMODs, as well as all the LMODs with cross-zone subentries, are listed.

**ZONESET**

indicates that SMP/E should list all ZONESET entries or the specified ZONESET entries.

**Note:** ZONESET is allowed when the SET command specifies the global zone.

For additional information on listing a specific entry type, see the section for that entry in Chapter 35. The description of the data in the entry, as well as examples for using the LIST command, are contained there.

**Syntax Notes**

1. Except where noted, you can specify multiple operands on a single LIST command. In comparison with specifying the operands on separate LIST commands, the overall performance of SMP/E is improved.
2. The order in which the entries are listed depends on their order in the SMP/E data set being used, not on the order specified on the LIST command.
3. You can mix mass-mode and select-mode requests on the same LIST command. For example:

```

SET      BDY(TGT1)      /* Set to target zone.      */
LIST     MOD            /* List all modules,        */
         MAC(MAC01     /* only two macros,        */
          MAC02)      /*                          */
         SRC(SRC01     /* only two source,        */
          SRC02)      /*                          */
         DLIB          /* all DLIBs,              */
         DDDEF         /* all DDDEFs,             */
         SYSMOD(UZ00001 /* only these five SYSMODS. */
          UZ00002 /*                          */
          UZ00003 /*                          */
          UZ00004 /*                          */
          UZ00005) /*                          */

```

4. If a given SYSMOD is specified on the SYSMODS operand, the SYSMOD is listed, regardless of whether it would be included or excluded by other operands.

### Data Sets Used

The following data sets may be needed to run the LIST command. They can be defined by DD statements or, preferably, by DDDEF entries. See Chapter 34 for more information about these data sets.

SMP_CNTL	SMPLOG	SMPOUT	SMPSNAP
SMP_CSI	SMPLOGA	SMPRPT	<i>zone</i>
SMP_LIST	SMPPTS	SMPSCDS	

#### Notes:

1. SMPPTS is required only if the MCS operand is specified.
2. SMPSCDS is required only if the BACKUP operand is specified.
3. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, SMP/E dynamically allocates the data sets using the ZONEINDEX information in the GLOBALZONE entry.

### Usage Notes

1. When the ALLZONES operand is specified, SMP/E displays the GLOBALZONE entry first, followed by all the entries within the global zone. Then, each zone defined by a ZONEINDEX subentry is processed in alphanumeric sequence by zone name. If SMP/E is unable to allocate the VSAM data set containing a particular zone, no further processing is done for that zone. Processing continues with the next zone.
2. Because SMP/E always opens the global zone for all processing, if there is an error during an attempt to open the global zone, you cannot process any SMP/E commands. Therefore, you cannot obtain a list of error messages from the SMPLOG data set.  
  
If you want to list the SMPLOG, and the CSI is damaged so that it cannot be opened, define a temporary CSI data set and use it to list the SMPLOG.
3. For more information about each entry type, see Chapter 35.

### Output

The LIST command output for each entry type is illustrated in the relevant section of Chapter 35.

The following reports are produced during LIST processing:

- File Allocation report
- LIST Summary report

For descriptions of these reports, see Chapter 31.

## Examples

The following examples are provided to help you use the LIST command:

- “Example 1: List All the Entries in a Particular Zone”
- “Example 2: List All the Entries of a Particular Type”
- “Example 3: List Specific Entries”
- “Example 4: List Entries Applicable to Specific FMIDs” on page 232
- “Example 5: Check Which SYSMODs Are Received But Not Installed” on page 232
- “Example 6: Check Whether SYSMODs Are Installed in the Related Zone” on page 232
- “Example 7: Compare the SYSMODs Installed in Two Zones of the Same Type” on page 233

**Note:** You can also use the REPORT SYSMODS command to compare zones. Besides telling you which SYSMODs are installed in one zone but not in another, REPORT SYSMODS also indicates which of the uninstalled SYSMODs are applicable to the second zone and generates commands you can run to install the SYSMODs in the second zone. For more information, see Chapter 18.

For examples of LIST commands and related output for each entry type, see Chapter 35.

### Example 1: List All the Entries in a Particular Zone

Suppose you have initialized the global zone with the definition entries needed to process that zone, and you want to list all those entries to check them. To do this, use the LIST command without any operands, as shown below:

```
SET      BDY(GLOBAL)      /* Set to global zone.    */
LIST     /* List all entries.  */
```

### Example 2: List All the Entries of a Particular Type

Suppose you want to check all the DDDEF entries defined in target zone TGT1. To do this, use the LIST command with just the DDDEF operand, as shown below:

```
SET      BDY(TGT1)        /* Set to target zone    */
LIST     DDDEF            /* List all DDDEF entries. */
```

If you want to check all the DDDEF entries defined in the global zone and all the zones defined to that global zone, add the ALLZONES operand to the LIST command shown above. (The global zone or any zone defined to it can be specified on the SET command.)

### Example 3: List Specific Entries

Suppose you encounter a problem on your system and need to determine whether SYSMOD UR12345 has been installed in target zone TGT1. To do this, use the LIST command with the SYSMOD operand and the SYSMOD ID, as shown below:

```
SET      BDY(TGT1)        /* Set to target zone.    */
LIST     SYSMOD(UR12345) /* List this SYSMOD.     */
```

If you want to check for that SYSMOD in the all the zones defined to the global zone, add the ALLZONES operand to the LIST command shown above. (The global zone or any zone defined to it can be specified on the SET command.)

#### Example 4: List Entries Applicable to Specific FMIDs

Suppose you need to determine whether target zone TGT1 contains entries for any elements owned by function SYSMOD HXY1100. To do this, use the LIST command with the FORFMID operand, as shown below:

```
SET      BDY(TGT1)           /* Set to target zone.    */
LIST     FORFMID(HXY1100)   /* List all entries       */
                        /* for this FMID.         */
```

#### Example 5: Check Which SYSMODs Are Received But Not Installed

Suppose you have received service into the global zone and are in the process of installing the service on your system. You want to see which of the SYSMODs you have received have not yet been installed in target zone TGT1. To do this, use the LIST command with the NOAPPLY operand and the zone name, as shown below:

```
SET      BDY(GLOBAL)        /* Set to global zone.    */
LIST     SYSMODS            /* List the SYSMODs that */
                        /* have been received but */
                        /* that have not been     */
                        /* applied to TGT1.       */
```

To see which of the SYSMODs you have received have not yet been installed in distribution zone DLIB1, use the LIST command with the NOACCEPT operand and the zone name, as shown below:

```
SET      BDY(GLOBAL)        /* Set to global zone.    */
LIST     SYSMODS            /* List the SYSMODs that */
                        /* have been received but */
                        /* that have not been     */
                        /* accepted in DLIB1.     */
```

#### Example 6: Check Whether SYSMODs Are Installed in the Related Zone

Suppose you need to compare the service level of target zone TGT1 with that of its related distribution zone, DLIB1. To do this, use the LIST command with the NOACCEPT operand, as shown below. (Because you are checking the related zone, you do not need to specify the zone name.)

```
SET      BDY(TGT1)          /* Set to TGT1.          */
LIST     SYSMODS            /* List the SYSMODs that */
                        /* have been applied but */
                        /* that have not been     */
                        /* accepted in the related */
                        /* zone.                   */
```

To see whether any SYSMODs have been installed in DLIB1 but not in TGT1, use

the LIST command with the NOAPPLY operand, as shown below:

```

SET      BDY(DLIB1)          /* Set to DLIB1.          */.
LIST     SYSMODS             /* List the SYSMODs that  */
                               /* have been accepted but  */
                               /* that have not been     */
                               /* applied to the related  */
                               /* zone.                  */.
NOAPPLY

```

## Example 7: Compare the SYSMODs Installed in Two Zones of the Same Type

Suppose you need to compare the service level of your production system and your test system, and you want to see which of the SYSMODs on the test system are not yet installed on the production system. To compare the target zones of the two systems, use the LIST command with the NOAPPLY operand, as shown below:

```

SET      BDY(TGT1)          /* Set to test zone TGT1. */.
LIST     SYSMODS             /* List the SYSMODs that  */
                               /* have been applied to    */
                               /* TGT1 but not to        */
                               /* production zone TGT2.  */.
NOAPPLY(TGT2)

```

To compare the distribution zones of the two systems, use the LIST command with the NOACCEPT operand and the test system zone name, as shown below:

```

SET      BDY(DLIB1)          /* Set to test zone DLIB1. */.
LIST     SYSMODS             /* List the SYSMODs that  */
                               /* have been accepted in   */
                               /* DLIB1 but not in       */
                               /* production zone DLIB2.  */.
NOACCEPT(DLIB2)

```

## Processing

Before SMP/E lists any entries, it first determines what type of LIST processing has been requested:

- If no entry names or entry types have been specified, SMP/E does mass-mode processing—for example, if `LIST` or `LIST ALLZONES` is specified. See “Mass-Mode Processing” on page 234 for a description of mass-mode processing.

If entry types without entry names are specified, SMP/E does mass-mode processing—for example, if `LIST MAC MOD` is specified to list all the MAC and MOD entries in a target zone.

- If entry names are specified, SMP/E does select-mode processing—for example, if `LIST MAC(MACA,MACB) MOD(MODA,MODB)` is specified to list specific MAC and MOD entries in a target zone. See “Select-Mode Processing” on page 234 for a description of select-mode processing.

**Note:** A single LIST command may combine mass-mode and select-mode processing.

## Mass-Mode Processing

In mass-mode processing, SMP/E checks whether any entry types have been specified. If so, SMP/E lists all entries of each specified type it finds in the set-to zone. SMP/E reads sequentially through the set-to zone. For each entry it finds, it checks whether the entry is of the specified type and whether the entry meets any other criteria specified (such as SYSMOD, MCS, FORFMID, and HOLDDATA). If the entry meets all the requirements, SMP/E formats and prints the data.

If no entry types are specified, SMP/E lists all the entries in the set-to zone. SMP/E reads sequentially through the set-to zone. For each entry it finds, it formats and prints the data.

## Select-Mode Processing

In select-mode processing, SMP/E lists all the explicitly specified entries that were found in the set-to zone. SMP/E goes directly to each specified entry and locates the data for the entry. For each entry it finds, it formats and prints the data.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of LIST processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

**Note:** Either the target zone or the distribution zone is used during initialization, according to the zone type specified in the previous SET command.

### 2. LIST processing

Global zone	—	Read with shared enqueue.
SMPPTS	—	Read with shared enqueue.
Target zone	—	Read with shared enqueue.
DLIB zone	—	Read with shared enqueue.

**Note:** The zones used depend on the LIST command operands and the zone type specified in the previous SET command.

### 3. Termination

All resources are freed.

---

## Chapter 11. The LOG Command

During SMP/E processing, messages recording what has occurred are written to the SMPOUT data set. Critical messages are also written to the SMPLOG data set to provide a permanent record of processing. Still other messages are written to SMPLOG to record internal processing, such as updating a SYSMOD entry or deleting an MTSMAC entry. In addition to the messages written by SMP/E, you may want to store messages in the SMPLOG, such as why a SYSMOD is being installed and who is installing it. You can do this using the LOG command.

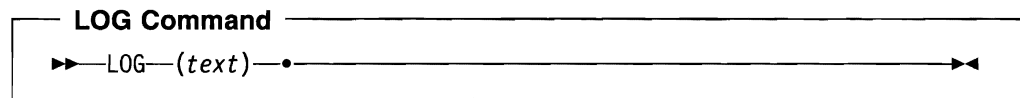
---

### Zones for SET BOUNDARY

Each zone should have its own SMPLOG data set, which should be defined by a DDDEF entry in that zone. If you want to use the LOG command to update a particular SMPLOG data set, the SET BOUNDARY command must specify the zone containing the DDDEF entry for that data set.

---

### Syntax



### Operands

*text*

represents the text that is to be written to the SMPLOG.

- LOG text may be in single-byte characters (such as English alphanumeric characters) or double-byte characters (such as Kanji).
- You can enter up to 250 bytes of data on a single LOG command, including blanks. For double-byte data, the 250-byte maximum includes all shift-in and shift-out characters as well as the double-byte characters. If you enter more than 250 bytes of data on one LOG command, the text is truncated. If you need to enter more than the maximum number of characters, use two or more LOG commands.
- SMP/E compresses the text to eliminate consecutive blanks.
- The text may span multiple lines.
- If parentheses are used in the text, they must be matched pairs.
- The format of the text stored in the SMPLOG data set may not be exactly as entered on the LOG command. For more information, see "Processing" on page 237.

---

## Data Sets Used

The following data sets may be needed to run the LOG command. They can be defined by DD statements or, preferably, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOG	SMPOUT	<i>zone</i>
SMP_CSI	SMPLOGA	SMPSNAP	

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated by use of the ZONEINDEX information in the GLOBALZONE entry.

---

## Output

The File Allocation report is produced during LOG processing. For a description of this report, see Chapter 31.

---

## Examples

The following examples are provided to help you use the LOG command:

- “Example 1: Writing a Message”
- “Example 2: Coding Parentheses Correctly” on page 237
- “Example 3: Listing an SMPLOG Data Set” on page 237

### Example 1: Writing a Message

You can use the following commands to write a message to three SMPLOG data sets associated with the global zone, target zone MVSTST1, and distribution zone MVSDLB1:

```
SET      BDY(GLOBAL)          /* Process global zone.      */.
LOG      (THIS MESSAGE WILL GO TO THE SMPLOG ASSOCIATED
          WITH THE GLOBAL ZONE).
SET      BDY(MVSTST1)        /* Process MVSTST1 tgt zone. */.
LOG      (THIS MESSAGE WILL GO TO THE SMPLOG ASSOCIATED
          WITH TARGET ZONE MVSTST1).
SET      BDY(MVSDLB1)        /* Process MVSDLB1 DLIB zone. */.
LOG      (THIS MESSAGE WILL GO TO THE SMPLOG ASSOCIATED
          WITH DISTRIBUTION ZONE MVSDLB1).
```

This example assumes you are having SMP/E dynamically allocate the SMPLOG DD statement; that is, no SMPLOG DD statement was provided in the JCL, and each zone contains a DDDEF entry for the SMPLOG. As SMP/E processes each SET command, SMP/E dynamically frees the SMPLOG data set currently allocated, and then dynamically allocates the SMPLOG DD statement, now pointing to the SMPLOG data set for the appropriate zone. SMP/E then writes the message to that SMPLOG data set.



## Example 2: Coding Parentheses Correctly

The following set of SMP/E commands has an error in the LOG command:

```
SET      BDY(GLOBAL)          /* Process global zone.    */.
LOG      (I AM ABOUT TO RECEIVE SERVICE LEVEL
          9301 IN ORDER TO INSTALL A NEW FUNCTION.
          (FUNCTION IS JXX1112)).
RECEIVE  SOURCEID(PUT9301) /* Receive service level.  */.
```

The LOG command does not have matched parentheses. The string **(FUNCTION IS JXX1112)** is in parentheses within the LOG command, but no closing parenthesis is found for the LOG command itself. SMP/E continues scanning the SMP\_CNTL input, looking for the closing parenthesis. Because it is not found, and because (we assume) there are no more commands, SMP/E issues an error message, and the message is not written to the SMPLOG.

Assume you had made an additional mistake in the RECEIVE command as follows:

```
SET      BDY(GLOBAL)          /* Process global zone.    */.
LOG      (I AM ABOUT TO RECEIVE SERVICE LEVEL
          9301 IN ORDER TO INSTALL A NEW FUNCTION.
          (FUNCTION IS JXX1112)).
RECEIVE  SOURCEID9PUT9301) /* Receive service level.  */.
```

Here, a 9 was mistakenly typed instead of a (. SMP/E finds the matching parenthesis after the SOURCEID operand, considers it to be the one for the LOG command, and would consider the text closed. Because no other data is found after the closing parenthesis (other than the period and a valid command comment) SMP/E writes the RECEIVE command to the SMPLOG as part of the LOG command text.

## Example 3: Listing an SMPLOG Data Set

The following is an example of listing the SMPLOG for the global zone:

```
SET      BDY(GLOBAL)          /* Process global zone.    */.
LIST     LOG                  /* List total log.        */.
```

In addition, parts of the SMPLOG can be listed by specifying a date range, as follows:

```
SET      BDY(GLOBAL)          /* Process global zone.    */.
LIST     LOG                  /* List part of SMPLOG    */.
          (06 01 94,         /* from 06/01/94 through  */.
          07 01 94).         /* 07/01/94.              */.
```

For additional information on listing the SMPLOG data set, see Chapter 10.

## Processing

When SMP/E processes the LOG command, the text from the command is placed in an internal buffer, either until the end of text is encountered or until the buffer is full.

As SMP/E is moving the text from the LOG command to the buffer area, it compresses the text by removing consecutive blanks. Thus, the format in which the

text is entered in the LOG command (such as spacing or number of lines) is not the same as when you list the LOG (using the LIST LOG command).

After SMP/E has finished scanning the LOG command, the buffer that was built is written to the SMPLOG data set. As each message is written, the current date and time are added to the message text so you can later list the SMPLOG for a specified date range.

---

## Chapter 12. The RECEIVE Command

RECEIVE is the first SMP/E command used to process any SYSMOD. It reads data from tape files or DASD data sets into the global zone, the SMPPTS, and temporary data sets (SMPTLIBs) for later SMP/E processing. The RECEIVE command processes data from two sources:

- The **SMPPTFIN** data set, which contains the modification control statements (MCSs) defining the SYSMODs, as well as any related ++ASSIGN statements.
- The **SMPHOLD** data set, which contains exception SYSMOD data (++HOLD and ++RELEASE statements).

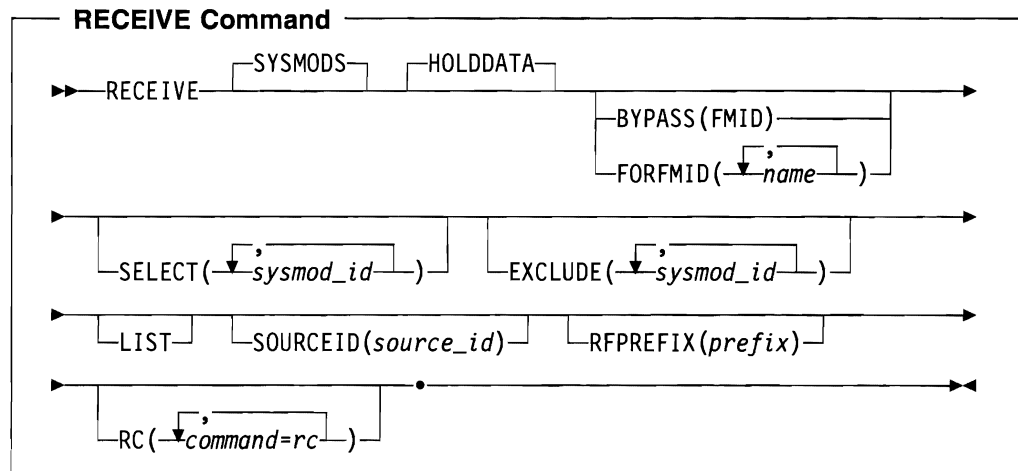
---

### Zones for SET BOUNDARY

For the RECEIVE command, the SET BOUNDARY command must specify the global zone.

---

### Syntax



### Operands

#### BYPASS(FMID)

indicates that all SYSMODs and HOLDDATA should be received, even if the associated FMID is not yet defined in the global zone.

**Note:** BYPASS is mutually exclusive with FORFMID.

#### EXCLUDE

specifies one or more SYSMOD IDs that should not be received.

**Note:** EXCLUDE can also be specified as E.

#### FORFMID

indicates that only SYSMODs and HOLDDATA for the specified FMIDs or FMIDSETs should be received. Any FMIDSETs specified must already be defined in the global zone.

## Notes:

1. FORFMID is mutually exclusive with BYPASS(FMID).
2. You cannot use the FORFMID operand as a substitute for UCLIN to add an FMID or FMIDSET to the global zone.
3. You can use FORFMID to receive a given function SYSMOD, along with other SYSMODs that are applicable to that function. For example, if function HXY1100 has not yet been received, you can use FORFMID(HXY1100) to receive that function plus the applicable SYSMODs.

SMP/E adds the FMID of the function to the global zone when it receives the function. As a result:

- Any SYSMODs that are applicable to the function and come **before** the function in the SMPPTFIN input stream will **not** be received, because the FMID of the function is not yet in the global zone.
  - Any SYSMODs that are applicable to the function and come **after** the function in the SMPPTFIN input stream **can** be received, because the FMID of the function is now in the global zone.
4. FORFMID does not affect SYSMODs specified on the SELECT operand.

## HOLDDATA

indicates that applicable data from SMPHOLD should be received.

- If you specify SELECT or FORFMID, HOLDDATA is received only for the SYSMODs included by those operands. For details, see “Example 4: Receiving Selected SYSMODs and HOLDDATA” on page 248.
- If you specify neither SELECT nor FORFMID, HOLDDATA is received for all FMIDs defined in the global zone.

## LIST

indicates that SMP/E should list the MCSs for each applicable SYSMOD.

## RC

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the RECEIVE command.

Before SMP/E processes the RECEIVE command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the RECEIVE command. Otherwise, the RECEIVE command fails. For more information about the RC operand, see Appendix D.

## Notes:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the RECEIVE command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

**RFPREFIX**

specifies the data set prefix used to construct the full data set name for RELFILE data sets when a SYSMOD packaged in RELFILE format is being processed by the RECEIVE command. This operand should be used when the name of the RELFILE data set starts with a prefix not identified by the RFDSNPFX operand on the header MCS of the associated SYSMOD. (For example, you may need to use RFPREFIX if you have loaded RELFILES into DASD data sets and have specified your own high-level qualifier for those data sets.)

Unless SMP/E is informed otherwise, it assumes that the name of a RELFILE data set is *sysmod-id.Fn*, where *sysmod-id* is the ID of the associated SYSMOD and *n* is the file number of the relative file.

**Note:** The RFPREFIX can contain from 1 to 26 alphanumeric (uppercase A–Z, 0–9), national (\$, #, @), dash (-), or period (.) characters.

After constructing the RELFILE data set name, SMP/E calculates the length of the full data set name (including both the RFPREFIX and RFDSNPFX values). If the length of the full data set name exceeds 44 characters, processing stops for the RECEIVE command.

**Notes:**

1. The RFPREFIX operand is ignored unless the FILES operand is specified on the header MCS statement for the SYSMOD being processed (++APAR, ++FUNCTION, ++PTF, or ++USERMOD MCS).
2. If the RFDSNPFX operand is specified on the header MCS, the RFPREFIX value specified on the RECEIVE command precedes the value from the RFDSNPFX operand when the name of the RELFILE data set is constructed.
3. If the RELFILE data sets are on DASD or are on tape and cataloged, the RELFILE data set name must not match the name to be used for the SMPTLIB data sets. If these data sets have the same name, the SMPTLIB data sets cannot be allocated. The DSPREFIX value in the OPTIONS entry is used to specify the high-level qualifier for the SMPTLIB data set names. For a description of the OPTIONS entry, see "OPTIONS Entry (Global Zone)" on page 741.

**SELECT**

specifies one or more SYSMOD IDs that should be received. Each SYSMOD must be in the SMPPTFIN data set, and the associated FMID must be either defined in the global zone or received concurrently.

**Notes:**

1. **SELECT** can also be specified as **S**.
2. If both SYSMODs and HOLDDATA are being processed and a selected SYSMOD is not received, the associated HOLDDATA **can** be received.
3. If you specify HOLDDATA and do not specify SYSMODS, only the HOLDDATA for the selected SYSMODs is received. The SYSMODs themselves are not received.

## SOURCEID

specifies a 1- to 8-character source identifier to be assigned to the SYSMODs being received. SMP/E assigns this source ID to all the SYSMODs processed by this RECEIVE command.

### Notes:

1. SMP/E does not check whether the source ID is unique. If the same value is specified on multiple RECEIVE commands, all SYSMODs processed by both commands have the same source ID.
2. If a source ID was specified for a particular SYSMOD on an ++ASSIGN statement in the input stream, that inline source ID is added to the source ID assigned to the SYSMOD by the RECEIVE command.

## SYSMODS

indicates that data from SMPPTFIN should be received.

**Note:** SYSMODS can also be specified as SYSMOD.

## Syntax Notes

1. The GLOBALZONE entry must contain at least one SREL value in order to receive either SYSMODs or exception SYSMOD data. If no SREL is defined, RECEIVE processing fails. If this happens, you can use UCLIN to add an SREL to the global zone, and then rerun the RECEIVE job.
2. When you specify SELECT with FORFMID, you can also specify EXCLUDE to exclude specific SYSMODs or HOLDDATA for the specified FMIDs.  
  
When you specify SELECT without FORFMID, however, there is no need to specify EXCLUDE.
3. If you specify neither SELECT, EXCLUDE, nor FORFMID, SMP/E considers all the data in SMPPTFIN and SMPHOLD eligible for processing.
4. SMP/E receives data from both SMPPTFIN and SMPHOLD if you specify either of the following:
  - Both SYSMODS and HOLDDATA
  - Neither SYSMODS nor HOLDDATA

If you specify only SYSMODS, HOLDDATA is excluded. If you specify only HOLDDATA, SYSMODs are excluded.

---

## Data Sets Used

The following data sets may be needed to run the RECEIVE command. They may be defined by DD statements or, normally, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPRPT	SYSUT1	<i>zone</i>
SMP_CSI	SMP_OUT	SMP_SNAP	SYSUT2	
SMP_HOLD	SMPPTFIN	SMP_TLIB	SYSUT3	
SMPLOG	SMPPTS	SYSPRINT		

**Notes:**

1. For RECEIVE processing, the SMPCSI DD statement refers to the data set containing the global zone, which is where SMP/E stores information about those SYSMODs received.
2. The SMPHOLD DD statement is required only if exception SYSMOD data is to be received from SMPHOLD.
3. The SMPPTFIN DD statement is required only if SYSMODs or ++ASSIGN statements are to be received from SMPPTFIN.
4. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, SMP/E dynamically allocates the data sets using the ZONEINDEX information in the GLOBALZONE entry.

---

**Usage Notes**

This section describes considerations for use of the following:

- “Receiving SYSMODs Packaged in Relative Files”
- “Defining an Installation-Wide Exit Routine for RECEIVE Processing” on page 246

**Receiving SYSMODs Packaged in Relative Files**

SMP/E can receive SYSMODs packaged in relative file format. These relative files can be on either DASD or tape. When the files are on DASD, the data sets must be cataloged and can be either partitioned data sets or sequential data sets (as produced by the copy utility when unloading partitioned data sets).

If a SYSMOD being received is packaged in relative files, those files are loaded into temporary direct access data sets as part of RECEIVE processing. (For more information about packaging SYSMODs in relative files, see the *Standard Packaging Rules for MVS-Based Products* manual.) These temporary data sets, called SMPTLIBs, can be allocated before you receive the SYSMOD, or they can be dynamically allocated during RECEIVE processing. (For details on defining SMPTLIB data sets, see “SMPTLIB Data Set” on page 618.)

If you want to allocate the SMPTLIBs yourself, you must let SMP/E know where to find the data sets. There are several ways you can do this:

- **Catalog:** You can catalog the SMPTLIB data sets, as well as allocate them. SMP/E allocates the data sets through the catalog when trying to find the data sets.
- **DDDEF entry:** Create a DDDEF entry named SMPTLIB in the global zone, and specify the following information:
  - **VOLUME**, to indicate the DASD volume or volumes on which the data sets reside.
  - Do not specify an initial or final disposition. SMP/E determines the appropriate values for RECEIVE processing.

- **DD statement:** Include an SMPTLIB DD statement in the RECEIVE JCL, and specify the following information:
  - **VOL**, to indicate the DASD volume or volumes on which the data sets reside.
  - **UNIT**, to indicate the unit on which the data sets reside.
  - Do not specify an initial or final disposition. SMP/E determines the appropriate values for RECEIVE processing.

**Note:** If the unit for the SMPTLIB data sets is not defined in SYSALLDA, you must use a DDDEF entry instead of a DD statement to define the data sets. Otherwise, the data sets are not allocated.

**Notes:**

1. If an error occurs when SMP/E is unloading the relative files or allocating SMPTLIBs during RECEIVE processing, preallocated SMPTLIBs **do** get deleted.
2. If the SMPTLIBs are not cataloged, SMP/E automatically catalogs them, and uncatalogs the data sets when it deletes them.

If you want SMP/E to dynamically allocate the SMPTLIBs, you can specify the information it needs on the SMPTLIB DD statement, in the SMPTLIB DDDEF entry, in the OPTIONS entry used for RECEIVE processing, or in some combination of these. Table 17 on page 245 shows what information SMP/E needs, and where it can be specified. (SMP/E checks the sources of information in the order shown.) Do not specify an initial or final disposition. SMP/E determines the appropriate values for RECEIVE processing.

**Notes:**

1. If you use a STEPCAT or JOBCAT DD statement, SMP/E catalogs the SMPTLIB data sets in the first catalog in the concatenation. Therefore, you must make sure that the high-level qualifier of the DSPREFIX value being used is compatible with that catalog. This is true for both preallocated data sets and dynamically allocated data sets.
2. For recommendations on the space required for a particular product's SMPTLIB data sets, see the installation documentation for that product. Either allocate the data sets yourself with this space, or specify it as indicated in Table 17 on page 245 if the SMPTLIB data sets are being dynamically allocated.
3. If you are not using SMS to manage your storage, then to properly allocate the SMPTLIB data sets, you need to provide allocation information from the sources listed in Table 17 on page 245.

If you are using SMS to manage your storage, then depending on how SMS is implemented on your system, you may not need to provide allocation information from the sources listed in Table 17 on page 245. If information required by SVC 99 (the dynamic allocation routine) is not defined to either SMP/E or SMS (such as information about volumes or space allocation), allocation fails for the SMPTLIB data sets.

Because the necessary information can be provided outside of SMP/E (through SMS), SMP/E does not issue error messages if any of this information is not specified.



Table 17. Information Used to Dynamically Allocate SMPTLIBs

Information	How to Specify on the DD Statement	How to Specify in the DDDEF Entry	How to Specify in the OPTIONS Entry	Default
Volume	VOL From 1 to 5 volumes	VOLUME From 1 to 5 volumes	Not specified here	No default
Unit	UNIT (must be part of SYSALLDA)	UNIT	Not specified here	SYSALLDA is used as the unit type.
Space allocation	Not specified here	SPACE	DSSPACE	No default, unless GIMMPDFT has been updated. See "Module GIMMPDFT" on page 824 for details.
Data set prefix	Not specified here	DSPREFIX This is used in the data set name: <i>prefix.</i> <i>sysmodid.Fnnnn</i>	DSPREFIX This is used in the data set name: <i>prefix.</i> <i>sysmodid.Fnnnn</i>	No prefix is used in the data set name: <i>sysmodid.Fnnnn</i>

### Limitations on Using PDSEs with SMPTLIB Data Sets

#### If you do not use the Storage Management Subsystem...

If you are not using Storage Management Subsystem (SMS) to manage your storage, you can skip this section.

SMPTLIB data sets should not be allocated as PDSEs, because IEBCOPY does not support copying an unloaded PDS load library from tape to a PDSE load library on DASD.

When SMS is used to manage SMP/E data sets, SMP/E may attempt to allocate SMPTLIB data sets as PDSEs. This occurs when the SMS default for new allocations is set to PDSE in SYS1.PARMLIB member IGDSMSxx with the DSNTYPE(LIBRARY) parameter.

You can avoid this situation by doing one of the following:

1. Add DSNTYPE=PDS to the SMPTLIB DDDEF in the global zone. This is the preferred method of circumvention.
2. Change the DSNTYPE in IGDSMSxx to PDS.
3. Preallocate SMPTLIB data sets as PDSs, rather than allow SMP/E to allocate them dynamically.

4. Create or update your data class ACS filter routine so that SMPTLIB data sets are allocated as PDSs. For example, suppose you set your DSPREFIX for your SMPTLIB data sets to SMPETLIB. Your data class ACS filter routine should assign any data set with a high-level qualifier of SMPETLIB to a data class that sets DSNTYPE to PDS:

```
IF &HLQ = 'SMPETLIB' THEN
  SET &DATACLAS = 'PDSCLASS'
```

where SMPETLIB is the DSPREFIX for the SMPTLIB data sets, and PDSCLASS is a SMS DATACLAS that sets DSNTYPE=PDS.

## Defining an Installation-Wide Exit Routine for RECEIVE Processing

You can define an installation-wide exit routine that examines each MCS (including comments and text) as it is read in from the SMPPTFIN data set. The exit routine is activated after the first record is read from SMPPTFIN, and is deactivated when RECEIVE processing stops. The RECEIVE exit routine can do any of the following:

- Change the input record
- Skip the input record
- Insert a record
- Stop RECEIVE processing for the current SYSMOD
- Stop RECEIVE processing
- Stop SMP/E processing

For more information about coding this installation-wide exit routine, see “RECEIVE Exit Routine” on page 799.

---

## Output

This section describes the listings and reports produced during RECEIVE processing.

## Listings

SMP/E does not print the SYSMOD or exception MCSs during RECEIVE processing unless you specify the LIST operand on the RECEIVE command. If LIST is specified, the following occurs:

- If **SELECT** or **EXCLUDE** is specified, the MCSs for those SYSMODs either selected or not specifically excluded are written to SMPOUT.
- If a SYSMOD is not applicable, no MCSs are listed for that SYSMOD.

**Note:** For SYSMODs with construction errors, SMP/E lists the MCSs up to the point of the error.

- Header MCSs (that is, ++FUNCTION, ++PTF, ++APAR, ++USERMOD) whose SYSMOD ID does not appear in the first record are written to SMPOUT, even though the SYSMOD was not selected or was excluded.

## Reports

Three reports are produced during RECEIVE processing:

- File Allocation report
- RECEIVE Exception SYSMOD Data report
- RECEIVE Summary report

For descriptions of these reports, see Chapter 31.

## Examples

The following examples are provided to help you use the RECEIVE command:

- “Example 1: Receiving SYSMODs and HOLDDATA”
- “Example 2: Receiving HOLDDATA Only” on page 248
- “Example 3: Receiving SYSMODs Only” on page 248
- “Example 4: Receiving Selected SYSMODs and HOLDDATA” on page 248
- “Example 5: Receiving SYSMODs and HOLDDATA for a Specific FMID” on page 248
- “Example 6: Receiving RELFILEs from DASD” on page 249

### Example 1: Receiving SYSMODs and HOLDDATA

Periodically, you must install service and process the related HOLDDATA for functions on your system. IBM supplies you with service and HOLDDATA on CBPDO tapes and ESO tapes. Both of these contain service and HOLDDATA, plus ++ASSIGN statements, which assign source IDs for PTFs that have been received.

When installing products and service, the first step is to receive the service and HOLDDATA into your SMP/E data sets. Here is a sample job that receives SYSMODs and HOLDDATA from a CBPDO and assigns them a source ID of PDO9324:

```
SET      BDY(GLOBAL)      /* Process global zone.    */.
RECEIVE SOURCEID(PDO9324) /* Receive SYSMODs and
                           assign SOURCEID.    */.
```

Besides receiving SYSMODs and HOLDDATA from the tape, SMP/E assigns a source ID of PDO9324 to the SYSMODs it has just received. In addition, it assigns the source IDs specified on the ++ASSIGN statements to SYSMODs that were just received or that had been previously received.

These source IDs can be specified later on the APPLY and ACCEPT commands to limit which SYSMODs should be installed.

## Example 2: Receiving HOLDDATA Only

If you do not want to keep the SYSMODs from your service tape in the SMPPTS until you are ready to install them, you should ensure that SMP/E has at least the exception SYSMOD information contained on that tape. This is important, as the information could affect some of those SYSMODs that are present on the SMPPTS and prevent inadvertent application of a PE-PTF. To process only the exception SYSMOD data, the following commands can be used:

```
SET      BDY(GLOBAL)      /* Process global zone.    */
RECEIVE  HOLDDATA        /* Receive HOLDDATA.
                               Note no SOURCEID is
                               assigned to HOLDDATA.    */
```

## Example 3: Receiving SYSMODs Only

If you choose to receive **only** exception data from the service tape (as in Example 2), you must receive those SYSMODs when you are ready to actually install them (during APPLY or ACCEPT command processing). This can be done by specifying the following commands:

```
SET      BDY(GLOBAL)      /* Process global zone.    */
RECEIVE  SYSMODS          /* Receive SYSMODs only
                               and assign
SOURCEID(MYSRCID) /* SOURCEID.                */
```

The SOURCEID operand assigns each of the SYSMODs received a SOURCEID value of PUT9301 that you can use during APPLY or ACCEPT to assist in selecting SYSMODs.

## Example 4: Receiving Selected SYSMODs and HOLDDATA

The RECEIVE command can be used to select individual SYSMODs or exception SYSMOD data applicable to selected SYSMODs. In this example, the commands cause SMP/E to receive two SYSMODs specified plus any exception SYSMOD data applicable to the SYSMODs:

```
SET      BDY(GLOBAL)      /* Process global zone.    */
RECEIVE  S(UZ12345,UZ12346) /* Receive selected
SYSMODS          /* SYSMODs
HOLDDATA          /* and HOLDDATA applicable
                               to them,
SOURCEID(MYSRCID) /* and assign a SOURCEID.    */
```

**Note:** The SYSMODs and HOLDDATA operands can be left off the RECEIVE command; processing is the same.

## Example 5: Receiving SYSMODs and HOLDDATA for a Specific FMID

You may want to receive SYSMODs and HOLDDATA for a particular FMID or FMIDSET. The following commands receive SYSMODs and HOLDDATA for the specified FMID, plus the selected SYSMOD:

```
SET      BDY(GLOBAL)      /* Process global zone.    */
RECEIVE  FORFMID(HXP1400) /* Receive SYSMODs and
                               HOLDDATA for HXP1400,
S(UZ12345)          /* plus selected SYSMOD.    */
```

## Example 6: Receiving RELFILES from DASD

Suppose a PTF packaged in RELFILE format (UR00001) has been shipped to you electronically, and you have read the RELFILES into DASD data sets. The high-level qualifier you used for the data set names is MYHLQ; the rest of the data set name follows the naming convention required by SMP/E. For example, the name of the data set containing RELFILE 1 is 'MYHLQ.UR00001.F1'. Now you want to receive the PTF from the DASD data sets. The following commands accomplish this task:

```
SET      BDY(GLOBAL)          /* Process global zone.    */.
RECEIVE SYSMODS              /* Receive SYSMODs        */.
        SOURCEID(IBMLINK)    /* and assign a SOURCEID. */.
        RFPREFIX(MYHLQ)     /* HLQ, RELFILES on DASD. */.
```

## Processing

RECEIVE processing includes these steps:

- Processing relative files
- Selecting SYSMODs
- Selecting ++HOLD and ++RELEASE statements
- Processing SYSMODs
- Processing ++ASSIGN statements
- Processing ++HOLD and ++RELEASE statements

## Processing Relative Files

SMP/E can receive SYSMODS packaged in relative file format from DASD, as well as tape. If SMP/E determines that the SMPPTFIN data set is located on tape, it assumes that the relative files are on tape. Otherwise, SMP/E assumes the RELFILES are on DASD and are cataloged. (If SMP/E assumes the RELFILES are on DASD but they actually are not, allocation fails for the RELFILES.)

If a SYSMOD being received is packaged in relative files, those files are loaded into temporary direct access data sets as part of RECEIVE processing. (For more information about packaging SYSMODs in relative files, see the *Standard Packaging Rules for MVS-Based Products* manual.) These temporary data sets, called SMPTLIBs, can be allocated before RECEIVE processing, or they can be dynamically allocated during RECEIVE processing. For more information, see "Receiving SYSMODs Packaged in Relative Files" on page 243. (Regardless of how the SMPTLIB data sets are allocated, they do not appear in the File Allocation report.) When the RELFILES are on DASD, SMP/E checks to ensure that the RELFILE data set name is not the same as the SMPTLIB data set name. If it is, RECEIVE processing stops.

To allocate SMPTLIB data sets, SMP/E checks the following, in the order shown:

1. **Catalog information:** SMP/E first tries to allocate the data set through the catalog. Therefore, if the SMPTLIB data set is preallocated and already cataloged, SMP/E uses that data set instead of allocating a new one.
2. **Volume information:** If the data set was not allocated through the catalog, but a volume list was specified in either the SMPTLIB DD statement or the SMPTLIB DDDEF entry, SMP/E searches the specified volumes to see whether the data set was preallocated on any of them.

- If so, SMP/E uses the preallocated data set instead of allocating a new one.
  - Otherwise, SMP/E tries to dynamically allocate a new data set on each specified volume using the parameters specified in the SMPTLIB DD statement or the SMPTLIB DDDEF entry (and the DSSPACE value in the current OPTIONS entry, if specified) until the data set is allocated.
3. **No catalog or volume information:** If no catalog or volume information is available to allocate the SMPTLIB data sets, SMP/E tries to dynamically allocate a new data set using the parameters specified in the SMPTLIB DD statement or the SMPTLIB DDDEF entry (and the DSSPACE value in the current OPTIONS entry, if specified).

### Dynamically Allocating SMPTLIB Data Sets

When dynamically allocating data sets, SMP/E uses information provided on the SMPTLIB DD statement, SMPTLIB DDDEF entry, and current OPTIONS entry. If there is a DD statement for the SMPTLIBs, SMP/E uses the volume specified on that statement. It then checks the SMPTLIB DDDEF entry to get additional information, such as space allocation parameters and the data set prefix. If SMP/E cannot find that information in the DDDEF entry, or if there is no SMPTLIB DDDEF entry, SMP/E gets the information from the OPTIONS entry that is in effect. (SMP/E always attempts to allocate data sets, even if no volume or space allocation information is provided.) If SMP/E finds a data set prefix, it assigns each SMPTLIB data set the name *prefix.sysmodid.Fnnnn*, where:

#### *prefix*

is defined by the DSPREFIX value in the DDDEF or OPTIONS entry. If there is no DSPREFIX value, no high-level qualifier is assigned.

#### **Notes:**

1. Any prefix value (RFDSNPFIX or RFPREFIX) specified for the associated RELFILE data set name is **not** included in the SMPTLIB data set name.
2. If the RELFILE data sets are on DASD or are on tape and are cataloged, the RELFILE data set name must not match the name to be used for the SMPTLIB data sets. If these data sets have the same name, the SMPTLIB data sets cannot be allocated. The DSPREFIX value in the OPTIONS entry is used to specify the high-level qualifier for the SMPTLIB data set names. For a description of the OPTIONS entry, see "OPTIONS Entry (Global Zone)" on page 741.

#### *sysmodid.Fnnnn*

is the data set name of the relative file being loaded. *sysmodid* is the ID of the associated SYSMOD, and *nnnn* is the relative file number of the file (for example, F1 or F2).

### Cataloging SMPTLIB Data Sets

SMP/E automatically catalogs SMPTLIB data sets that have been either dynamically allocated or preallocated, but not cataloged. It uncatalogs the SMPTLIB data sets when it deletes them.

**Note:** If a STEPCAT or JOBCAT DD statement is used, SMP/E catalogs the SMPTLIB data sets in the first catalog in the concatenation. Therefore, the high-level qualifier of the DSPREFIX value being used must be compatible

with that catalog. This is true for both preallocated data sets and dynamically allocated data sets.

### Loading the Relative Files

SMP/E uses IEBCOPY or a user-defined copy utility to load the relative files into the SMPTLIB data sets. (A user-defined utility is specified through OPTIONS and UTILITY entries.) Each element defined by an MCS in the SYSMOD is selectively copied, including each ALIAS, DALIAS, TALIAS, and MALIAS name. This selective copying ensures that the relative files contain the correct elements. If the copy is successful, any unused space in the SMPTLIB data sets is released.

**Note:** Any required aliases must be in the distribution libraries before the libraries are put into relative files. Relative files represent distribution library data sets for a function. When SMP/E loads the relative files, it copies in only members from the unloaded data sets; it does not cause aliases to be created for those members.

When a SYSMOD has been received and its relative files have been loaded, SMP/E saves the high-level qualifier for the SMPTLIB data sets as the TLIBPREFIX value in the global zone SYSMOD entry for that SYSMOD. This allows you to change the DSPREFIX value in the DDDEF or OPTIONS entry that was used without affecting any SYSMODs that have already been received.

Sometimes errors occur while the SMPTLIB data sets are being allocated or the relative files are being unloaded. In this case, the return code from IEBCOPY may be higher than the SMP/E default value or the maximum return code value defined in the COPY UTILITY entry that is in effect. If such an error occurs, SMP/E cannot receive the SYSMOD, and it deletes all the SMPTLIB data sets associated with that SYSMOD, even if they were preallocated.

## Selecting SYSMODs

A SYSMOD is selected if it meets all the following conditions:

- The SYSMOD was specified in the SELECT list. This is called *select mode*.
- Neither **SELECT** nor **EXCLUDE** was specified. This is called *mass mode*.
- The SYSMOD was not specified in the EXCLUDE list.
- The SYSMOD is a base function and is applicable to one of the SRELS (system releases) defined in the global zone.
- The SYSMOD is a service SYSMOD or a dependent function and is applicable to one of the SRELS and one of the FMIDs defined in the global zone.

**Note:** If **BYPASS(FMID)** was specified, SMP/E does not check whether the applicable FMID for service is already defined in the global zone. In this case, SMP/E selects all SYSMODs that are applicable to an SREL defined in the global zone and to all exception SYSMOD data.

- An entry does not already exist in both the SMPPTS and the global zone for the SYSMOD.

**Note:** If an entry exists in either the global zone or the SMPPTS, but not both, SMP/E assumes that an error occurred during a previous RECEIVE attempt. In this case, it selects the current SYSMOD and replaces any existing entries for that SYSMOD in the global zone or SMPPTS.

A SYSMOD is selected, regardless of the status of its requisites. These are checked and resolved later when the SYSMOD is applied and accepted.

Generally, a SYSMOD that has already been successfully received cannot be received again unless it is first rejected. However, SMP/E may automatically rereceive a SYSMOD if both of these conditions are met:

- The REWORK operand is coded on the ++PTF, ++FUNCTION, ++APAR, or ++USERMOD statement. Typically, SYSMODs with the REWORK operand have been reworked by IBM for minor changes.
- The REWORK level is higher than the level for the previous version of the SYSMOD.

This saves you from having to reject and receive again the SYSMODs yourself. For more information about the REWORK operand, see the descriptions of these MCSs in Chapter 32.

**Note:** If a SYSMOD appears more than once in the SMPPTFIN data set, the first occurrence may be received. However, none of the subsequent versions of the SYSMOD are received, even if their REWORK level is higher than the one for the first version of the SYSMOD. (Message GIM40001 is issued for each of the subsequent versions of the SYSMOD.)

### Selecting ++HOLD and ++RELEASE Statements

SMP/E determines which ++HOLD and ++RELEASE statements (collectively called HOLDDATA) to select from SMPHOLD according to whether the SELECT or FORFMID operand is specified.

- If SELECT or FORFMID is specified, SMP/E selects HOLDDATA that is applicable to the SYSMODs included by these operands.
- If neither SELECT nor FORFMID is specified, SMP/E selects HOLDDATA whose applicable FMID is defined in the global zone.

### Processing SYSMODs

For each SYSMOD selected for RECEIVE processing, SMP/E does the following:

- Saves the complete SYSMOD, unchanged and including any inline changes, as a member in the SMPPTS data set. This member is called an *MCS entry*.
- Creates a SYSMOD entry in the global zone. This entry contains information SMP/E needs to determine SYSMOD applicability during later SMP/E processing. For example, if a source ID was specified on the RECEIVE command, SMP/E saves that value in the SYSMOD entry.

**Notes:**

1. If a SYSMOD entry already exists, but contains only HOLD reason IDs, those reason IDs are saved in the SYSMOD entry for the SYSMOD that was received.



2. If a SYSMOD was received again, the existing SYSMOD entry in the global zone is completely replaced, except for the ACCID and APPID subentries. These are saved in the new SYSMOD entry so there is still a record of the zones where the SYSMOD has already been installed.

- Adds the FMID of each function SYSMOD received to the global zone. This enables SMP/E to receive SYSMODs applicable to that function SYSMOD.

MCS entries for SYSMODs that were not selected are not saved in the SMPPTS data set.

## Processing ++ASSIGN Statements

For each ++ASSIGN statement that was successfully processed, SMP/E associates the source ID with the specified SYSMODs. The source ID is assigned only to SYSMODs that are in both the global zone and the SMPPTS data set. If the same SYSMOD is specified on more than one ++ASSIGN statement, all the source IDs are associated with the SYSMOD. A source ID specified on a ++ASSIGN statement is added to any source ID that is assigned to a specified SYSMOD by the RECEIVE command. It is also added to any source IDs currently associated with a specified SYSMOD that has already been received.

## Processing ++HOLD and ++RELEASE Statements

All MCSs from SMPHOLD are processed in the order in which they occur. (++)HOLD contained within SYSMODs that were received are processed when those SYSMODs are processed.) For each ++HOLD and ++RELEASE statement that was selected, SMP/E does the following:

- For ++HOLD statements, SMP/E adds the reason IDs to the associated global zone SYSMOD entry. If no SYSMOD entry exists, SMP/E builds one that contains just the reason IDs. SMP/E also saves the ++HOLD statement in the global zone. This is called a *HOLDDATA entry*.

**Note:** For a given SYSMOD, SMP/E does not save multiple entries or sub-entries for a given reason ID. For example, assume SMP/E has already received the following ++HOLD statement:

```
++HOLD (UZ12345) FMID(FXY1040) SYSTEM REASON(DOC)
      COMMENT(message XXX123 was changed. enter U to reply.).
```

Later, SMP/E receives another ++HOLD statement for the same SYSMOD, hold type, and reason ID, but the comment is different:

```
++HOLD (UZ12345) FMID(FXY1040) SYSTEM REASON(DOC)
      COMMENT(default for xyz command changed to NO.).
```

Information from the second ++HOLD statement replaces the information from the first ++HOLD statement.

- For ++RELEASE statements, SMP/E removes the specified reason ID from the global zone SYSMOD entry and deletes the ++HOLD statement for that reason ID from the global zone.

### Zone and Data Set Sharing Considerations

The following indicates the phases of RECEIVE processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

1. Initialization

Global zone — Read without enqueue.

2. RECEIVE processing

Global zone — Update with exclusive enqueue.

SMPPTS — Update with exclusive enqueue.

3. Termination

All resources are freed.

---

## Chapter 13. The REJECT Command

The REJECT command allows you to manage the global zone and SMPPTS by deleting global zone SYSMOD entries, HOLDDATA entries, SMPPTS MCS entries, and any associated SMPTLIB data sets. REJECT is helpful if the SMPPTS is being used as a permanent database for all SYSMODs, including those that have been installed. You can also use it to purge old data from the global zone and SMPPTS.

To use the REJECT command, you must first determine which processing mode of the command you want to use. The mode you choose depends on the data you want to delete. These are the modes of REJECT processing:

- **Mass mode:** SMP/E rejects all SYSMODs that have been received but not installed. Generally, SMP/E rejects these SYSMODs only if they have been neither accepted nor applied. However, you can specify operands to prevent SMP/E from checking where the SYSMODs have been installed.
- **Select mode:** SMP/E rejects specific SYSMODs that have been received but not installed. Generally, SMP/E rejects these SYSMODs only if they have been neither accepted nor applied. However, you can specify operands to prevent SMP/E from checking where the SYSMODs have been installed.
- **PURGE mode:** SMP/E rejects all SYSMODs that have been accepted into the specified distribution zones. PURGE mode can be used when SYSMODs were not automatically deleted once they were accepted. This is the case if NOPURGE was coded in the OPTIONS entry used to process the distribution zone.
- **NOFMID mode:** SMP/E rejects all SYSMODs applicable to functions that are not part of the system. NOFMID mode can be used to delete service for all functions that have been deleted from the global zone.

REJECT command processing modes are mutually exclusive. No two modes can be used together.

---

### Zones for SET BOUNDARY

For the REJECT command, the SET BOUNDARY command must specify the global zone.

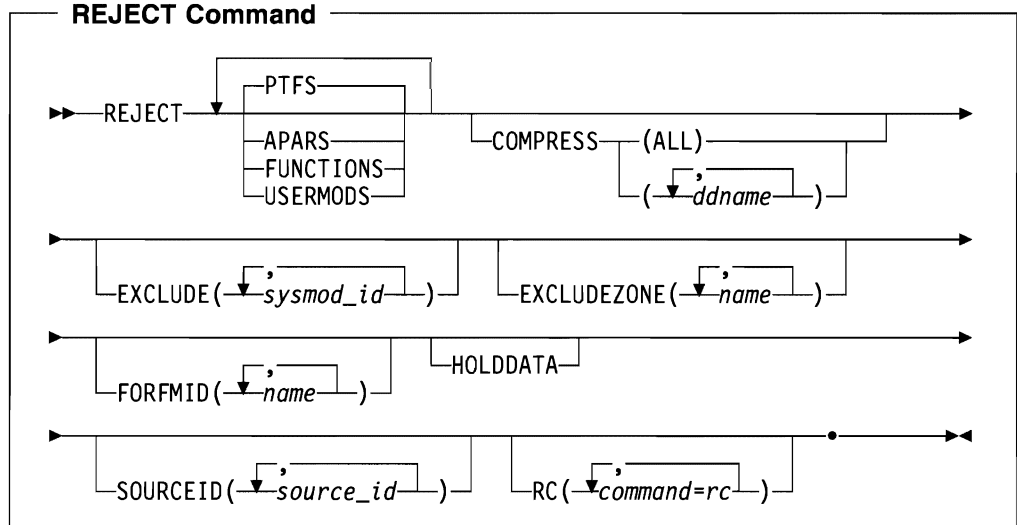
---

### Syntax

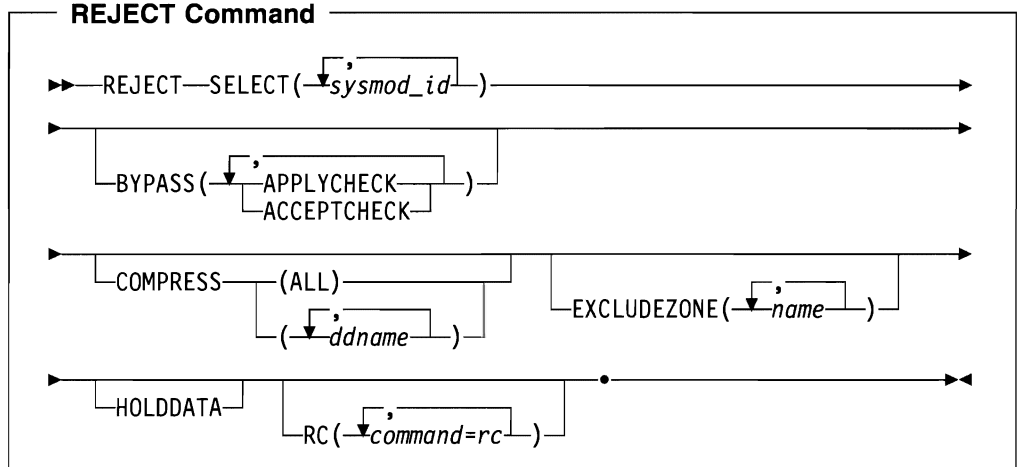
This section shows the syntax for the modes of REJECT processing:

- Mass mode
- Select mode
- PURGE mode
- NOFMID mode

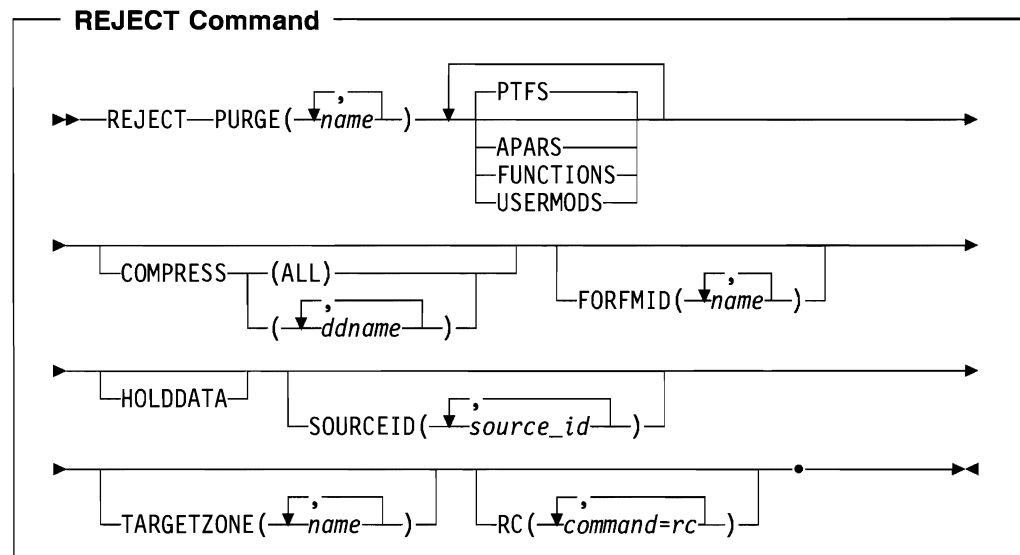
### Mass Mode Syntax



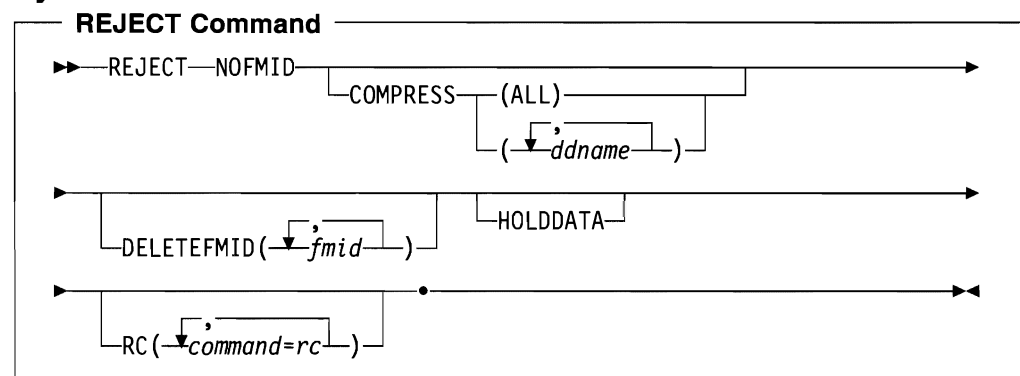
### Select Mode Syntax



## PURGE Mode Syntax



## NOFMID Mode Syntax



## Operands

### APARS

indicates that APARs should be rejected.

#### Notes:

1. APARS is allowed only in mass mode and PURGE mode.
2. APARS can also be specified as APAR.

### BYPASS

indicates that SMP/E should reject SYSMODs that have been installed.

### APPLYCHECK

indicates that selected SYSMODs should be rejected, even if they have been applied.

### ACCEPTCHECK

indicates that selected SYSMODs can be rejected, even if they have been accepted.

### Notes:

1. **BYPASS** is allowed only in select mode.
2. **APPLYCHECK** can also be specified as **APPCHK**.
3. **ACCEPTCHECK** can also be specified as **ACCCHK**.
4. To reject a superseded **SYSMOD**, you must either specify the appropriate **BYPASS** operands, or you must use the **EXCLUDEZONE** operand to specify the zones in which the **SYSMOD** is superseded.

### COMPRESS

indicates which partitioned data sets should be compressed.

- If you specify **ALL**, any partitioned data sets that were updated are compressed. In addition, the **SMPPTS** data set is compressed regardless of whether it was updated.
- If you specify one or more specific **ddnames**, only the data sets they apply to are compressed. Those data sets are compressed regardless of whether they were updated.

### Notes:

1. **COMPRESS** is allowed in all modes of **REJECT** processing.
2. **COMPRESS** can also be specified as **C**.

### DELETEFMID

specifies one or more **FMIDs** that are to be deleted from the **GLOBALZONE** entry before **SMP/E** selects the **SYSMODs** or **HOLDDATA** to be rejected.

### Notes:

1. **DELETEFMID** is allowed only in **NOFMID** mode.
2. **DELETEFMID** can also be specified as **DFMID**.

### EXCLUDE

specifies one or more **SYSMODs** that should not be rejected.

### Notes:

1. **EXCLUDE** is allowed only in mass mode.
2. **EXCLUDE** can also be specified as **E**.

### EXCLUDEZONE

indicates that **SMP/E** should not check whether **SYSMODs** have been installed in the specified zones or **ZONESETs**.

For each specified value, **SMP/E** first checks for a **ZONESET** with the same name. If none is found, **SMP/E** checks for a zone with the same name. For example, suppose you have a **ZONESET** named **MVS1** and a zone named **MVS1**. If you specify **MVS1** on this operand, **SMP/E** assumes you want to use the zones defined in **ZONESET MVS1** (which might or might not include zone **MVS1**), and not the individual zone **MVS1**.

**Notes:**

1. EXCLUDEZONE is allowed only in mass mode and select mode.
2. EXCLUDEZONE can also be specified as EZONE.
3. EXCLUDEZONE cannot specify all the zones defined by ZONEINDEX sub-entries. If you are doing select-mode processing and you do not want SMP/E to check any zones, you can specify **BYPASS(APPLYCHECK,ACCEPTCHECK)**. If you are doing mass-mode processing, there is no way to have SMP/E ignore all the zones.
4. You should be careful when using the EXCLUDEZONE operand. A SYSMOD that is installed in one of the excluded zones may be rejected, even if you were later going to install it in another zone. Because the rejected SYSMOD is no longer in the SMPPTS, it cannot be installed in any more zones.
5. To reject a superseded SYSMOD, you must either specify the appropriate BYPASS operands, or you must use the EXCLUDEZONE operand to specify the zones in which the SYSMOD is superseded.

**FORFMID**

indicates that only SYSMODs and HOLDDATA for the specified FMIDs or FMIDSETs should be rejected.

**Notes:**

1. FORFMID is allowed only in mass mode and PURGE mode.
2. If no SYSMOD types are specified, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.

**FUNCTIONS**

indicates that functions should be rejected.

**Notes:**

1. FUNCTIONS is allowed only in mass mode and PURGE mode.
2. FUNCTIONS can also be specified as **FUNCTION**.

**HOLDDATA**

indicates that SMP/E should reject HOLDDATA entries. There are two types of HOLDDATA entries: those that have an associated SYSMOD entry, and those that have no associated SYSMOD entry. [For more information, see "HOLDDATA Entry (Global Zone)" on page 695.] How SMP/E processes HOLDDATA entries depends on the mode of REJECT processing you choose.

**Note:** HOLDDATA is allowed in all modes of REJECT processing.

- **Mass mode:** If you specify **HOLDDATA**, SMP/E deletes HOLDDATA entries associated with the SYSMOD entries that are also being rejected. However, it does not delete any HOLDDATA entries that have no associated SYSMOD entries.

If you do not specify **HOLDDATA**, SMP/E does not delete any HOLDDATA entries.

- **Select mode:** If you specify **HOLDDATA**, SMP/E deletes HOLDDATA entries associated with the eligible SYSMOD IDs, regardless of whether an associated SYSMOD entry exists.

If you do not specify **HOLDDATA**, SMP/E does not delete any **HOLDDATA** entries.

- **PURGE mode:** If you specify **HOLDDATA**, SMP/E deletes **HOLDDATA** entries associated with the **SYSMOD** entries also being rejected. It also deletes **HOLDDATA** entries that have no associated **SYSMOD** entries but meet the conditions specified by other **REJECT** operands.

If you specify **FORFMID** or **SOURCEID**, SMP/E does not delete **HOLDDATA** entries that have no associated **SYSMOD** entries.

- **NOFMID mode:** If you specify **HOLDDATA**, SMP/E deletes **HOLDDATA** entries associated with the **SYSMOD** entries that are also being rejected. In addition, SMP/E deletes **HOLDDATA** entries whose associated **FMID** is not defined in the global zone.

If you do not specify **HOLDDATA**, SMP/E deletes **HOLDDATA** entries associated with the **SYSMOD** entries that are also being rejected. However, it does not delete any **HOLDDATA** entries whose associated **FMID** is not defined in the global zone.

### NOFMID

indicates that SMP/E is to reject **SYSMODs** applicable to functions that are not part of the system. (The **FMID** they apply to is not in the **GLOBALZONE** entry.)

If **DELETEFMID** is also specified, SMP/E deletes the specified **FMIDs** from the **GLOBALZONE** entry before determining which **SYSMODs** and **HOLDDATA** to delete.

**Note:** **NOFMID** is allowed only in **NOFMID** mode.

### PTFS

indicates that **PTFs** should be rejected. This is the default **SYSMOD** type operand. In mass or **PURGE** mode processing, if no **SYSMOD** type is specified, only **PTFs** are rejected.

#### Notes:

1. **PTFS** is allowed only in mass mode and **PURGE** mode.
2. **PTFS** can also be specified as **PTF**.

### PURGE

indicates that SMP/E should reject only **SYSMOD** and **HOLDDATA** entries for **SYSMODs** that have been accepted into the specified distribution zones or **ZONESETs**.

For each specified value, SMP/E first checks for a **ZONESET** with the same name. If none is found, SMP/E checks for a zone with the same name. For example, suppose you have a **ZONESET** named **MVS1** and a zone named **MVS1**. If you specify **MVS1** on this operand, SMP/E assumes you want to use the zones defined in **ZONESET MVS1** (which might or might not include zone **MVS1**), not the individual zone **MVS1**.

Any individual zones you specify must be distribution zones. Among all the zones and **ZONESETs** you specify, there must be at least one distribution zone.

- If you specify a single zone, SMP/E rejects entries only for **SYSMODs** that have been accepted into that zone.



- If you specify more than one zone (including zones in a ZONESET), SMP/E rejects entries only for SYSMODs that have been installed in at least one distribution zone and have been installed in all the distribution zones they apply to.

If **TARGETZONE** is also specified, SMP/E reject entries only for SYSMODs that have also been installed in the specified target zones where they are applicable.

HOLDDATA entries are rejected only if **HOLDDATA** was specified.

**Notes:**

1. PURGE is only allowed in PURGE mode.
2. PURGE cannot be used to reject **specific** SYSMODs that have been accepted. To do this you must specify **BYPASS(ACCCHK)** in select mode.

**RC**

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the REJECT command.

Before SMP/E processes the REJECT command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the REJECT command. Otherwise, the REJECT command fails. For more information about the RC operand, see Appendix D.

**Notes:**

1. The RC operand **must be the last** operand specified on the command.
2. RC is allowed in all modes of REJECT processing.
3. If you do specify the RC operand, return codes for commands not specified do not affect processing for the REJECT command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

**SELECT**

specifies one or more SYSMODs that should be rejected.

**Notes:**

1. SELECT is only allowed in select mode.
2. **SELECT** can also be specified as **S**.

**SOURCEID**

indicates that only entries for SYSMODs associated with the specified source IDs should be rejected.

**Notes:**

1. SOURCEID is allowed only in mass mode and PURGE mode.
2. There are two ways to specify source IDs:
  - Explicitly, by specifying a particular source ID (for example, **PUT9303**). In this case, only that particular source ID is used.

- Implicitly, by specifying either \* or c\* (for example, PUT\*), where c is a 1- to 7-character string. In the second case, all source IDs beginning with the specified character string are used.
3. A given source ID can be explicitly specified **only once** on the SOURCEID operand.
  4. If no SYSMOD types are specified, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.

### TARGETZONE

indicates that SMP/E should only reject SYSMOD and HOLDDATA entries for SYSMODs that have been applied to the specified target zones or ZONESETS.

For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. For example, suppose you have a ZONESET named MVS1 and a zone named MVS1. If you specify MVS1 on this operand, SMP/E assumes you want to use the zones defined in ZONESET MVS1 (which might or might not include zone MVS1), not the individual zone MVS1.

Any individual zones you specify must be target zones. Among all the zones and ZONESETs you specify, there must be at least one target zone.

SMP/E rejects entries for SYSMODs that have been installed in the specified target zones where they are applicable. If a SYSMOD is not applicable to any of the specified target zones, it may still be rejected if it is installed in the specified distribution zones where it is applicable.

#### Notes:

1. TARGETZONE is allowed only in PURGE mode.
2. TARGETZONE cannot be used to reject **specific** SYSMODs that have been applied. To do this, you must specify **BYPASS(APPCHK)** in select mode.

### USERMODS

indicates that USERMODs should be rejected.

#### Notes:

1. USERMODS is allowed only in mass mode and PURGE mode.
2. USERMODS can also be specified as **USERMOD**.

---

## Data Sets Used

The following data sets may be needed to run the REJECT command. They can be defined by DD statements or, normally, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPOUT	SMPSNAP	SYSUT1
SMP_CSI	SMPPTS	SMPTLIB	<i>zone</i>
SMPLOG	SMPRPT	SYSPRINT	
SMPLOGA			

**Note:** *zone* represents the DD statements required for each distribution zone used by this command. If the DD statements are not specified, the ZONEINDEX

information in the GLOBALZONE entry is used to dynamically allocate the data sets. *zone* is required if the PURGE operand is specified.

## Output

Output from the REJECT command includes reports, as well as statistics written to SMPOUT and SMPLOG.

## Reports

Two reports are produced during REJECT processing:

- File Allocation report
- REJECT Summary report

See Chapter 31 for descriptions of these reports.

## Statistics

SMP/E writes statistics to SMPOUT and SMPLOG to summarize what happened during REJECT processing. These statistics follow the message issued at the completion of REJECT processing. Figure 5 shows the format of the statistics.

```

REJECT STATISTICS

SYSMODS REJECTED - nnnnnn  SYSMODS NOT REJECTED - nnnnnn
FMIDS DELETED    - nnnnnn  FMIDS NOT DELETED    - nnnnnn
HOLDDATA DELETED - nnnnnn
  
```

Figure 5. REJECT Statistics

### SYSMODS REJECTED

is the number of SYSMODs that were rejected.

### SYSMODS NOT REJECTED

is the number of SYSMODs that were candidates but were not rejected. The reason appears in the REJECT Summary report.

### FMIDS DELETED

is the number of FMIDs that were deleted. This includes FMIDs specified on the DELETEDFMID operand in NOFMID mode or FMIDs that were deleted from the GLOBALZONE entry in other modes when functions were rejected.

### FMIDS NOT DELETED

is the number of FMIDs specified on the DELETEDFMID operand that were not deleted.

### HOLDDATA DELETED

is the number of external HOLDDATA entries that were deleted. This number does **not** include HOLDDATA entries (for system holds).

**Note:** Internal system HOLDDATA is contained within a SYSMOD and is considered part of the rejected SYSMOD. Therefore, internal HOLDDATA is not reported as deleted HOLDDATA.

## Examples

The following examples are provided to help you use the REJECT command:

- “Example 1: Rejecting All SYSMODs That Have Not Been Installed (Mass Mode)”
- “Example 2: Rejecting All SYSMODs for a Specific Function (Mass Mode)” on page 265
- “Example 3: Rejecting Selected SYSMODs That Have Been Applied (Select Mode)” on page 265
- “Example 4: Rejecting Selected SYSMODs That Have Been Accepted and Applied (Select Mode)” on page 265
- “Example 5: Rejecting HOLDDATA That Has No SYSMOD Entry (Select Mode)” on page 265
- “Example 6: Rejecting SYSMODs That Have Been Accepted (PURGE Mode)” on page 266
- “Example 7: Rejecting SYSMODs That Have Been Accepted and Applied (PURGE Mode)” on page 266
- “Example 8: Rejecting SYSMODs for Undefined Functions (NOFMID Mode)” on page 266
- “Example 9: Deleting Service for a Group of Source IDs” on page 267
- “Example 10: Rejecting Selected SYSMODs That Have Been Superseded (Select Mode)” on page 267

### Example 1: Rejecting All SYSMODs That Have Not Been Installed (Mass Mode)

Assume you want to delete all SYSMODs that have been received but not installed. You can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to global zone.    */
REJECT  APARS             /* Reject all received-only */
        FUNCTIONS        /* SYSMODs.                */
        PTFS
        USERMODS.
```

**Note:** Be careful when you use this format for REJECT. It may reject SYSMODs you wanted to keep, and you would have to receive them again.

If you want to reject only PTFs that have been received but not installed, you can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to global zone.    */
REJECT                                     /* Reject all received-only */
                                           PTFs.                    */
```

If no SYSMOD types are specified on a REJECT command for mass mode, SMP/E rejects only PTFs.

## Example 2: Rejecting All SYSMODs for a Specific Function (Mass Mode)

Assume you have received a new function (HMX1100), as well as service for that function. You have now decided to delete that function and all the associated service for it. You can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to global zone.      */
REJECT  APARS             /* Reject all APARs,        */
        FUNCTIONS        /* functions,                */
        PTFS             /* PTFs, and                 */
        USERMODS        /* USERMODs for             */
        FORFMID(HMX1100) /* HMX1100.                  */
```

## Example 3: Rejecting Selected SYSMODs That Have Been Applied (Select Mode)

Assume you have applied a specific user modification but have not accepted it. You want to reject the current version, update the SYSMOD, and then reapply it. You can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to global zone.      */
REJECT  S(MYMOD01)        /* Reject this SYSMOD       */
        BYPASS(           /* even though it was       */
        APPCHK)          /* applied.                  */
```

## Example 4: Rejecting Selected SYSMODs That Have Been Accepted and Applied (Select Mode)

Assume you have applied a user modification and accepted it (with NOPURGE in the OPTIONS entry). You want to reject the current version, update the SYSMOD, and then reapply and reaccept it. You can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to global zone.      */
REJECT  S(MYMOD01)        /* Reject this SYSMOD       */
        BYPASS(           /* even though it was       */
        ACCEPTCHECK /* accepted and              */
        APPLYCHECK) /* applied.                  */
```

## Example 5: Rejecting HOLDDATA That Has No SYSMOD Entry (Select Mode)

Assume you have received a ++HOLD statement for PTF UZ04356 from the SMPHOLD data set, but you have not yet received the PTF itself. There is a HOLDDATA entry but no SYSMOD entry. If you do not plan to install that PTF, you may want to delete the HOLDDATA entry. You can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to global zone.      */
REJECT  S(UZ04356)        /* For this SYSMOD,         */
        HOLDDATA         /* reject the HOLDDATA.     */
```

**Note:** If there had been a SYSMOD entry for UZ04356, these commands would have deleted the SYSMOD entry along with the HOLDDATA entry.

## Example 6: Rejecting SYSMODs That Have Been Accepted (PURGE Mode)

Assume you have been using your SMPPTS as a database for all SYSMODs (by using NOPURGE in the OPTIONS entry). You have been receiving service through ESO tapes, which assign the SYSMODs source IDs to identify the service levels you have installed. Now you want to purge PTFs from service levels 9307 through 9310 that have been accepted into distribution zone DLIB1. You can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to global zone. */.
REJECT  PURGE(DLIB1)     /* Reject SYSMODs installed
                           in this DLIB zone */
        SOURCEID(PUT9307, /* for these service levels.*/
                PUT9308, /* */
                PUT9309, /* */
                PUT9310) /* */.
```

### Notes:

1. Because no SYSMOD types were specified, only PTFs are rejected.
2. Without the SOURCEID operand, SMP/E rejects the SYSMOD entries for all the PTFs that had been accepted into DLIB1.

## Example 7: Rejecting SYSMODs That Have Been Accepted and Applied (PURGE Mode)

Assume you have a system with three target zones (TMV1, TMV2, and TMV3) and two DLIB zones (DMVA and DMVB). You have set up two ZONESETs to make it easier to maintain these zones. MVSSET contains TMV1, TMV2, and DMVA, and MVSTEST contains TMV3 and DMVB.

Assume you want to reject all the PTFs that were installed in the zones contained in the MVSSET ZONESET. You can use these commands:

```
SET      BDY(GLOBAL)      /* Process global zone. */.
REJECT  PURGE(MVSSET)     /* Reject PTFs in MVSSET */
        TZONE(MVSSET)     /* DLIB and target zones. */.
```

## Example 8: Rejecting SYSMODs for Undefined Functions (NOFMID Mode)

Assume you had received, applied, and accepted function HMX1101. The function was automatically deleted from the global zone and SMPPTS when it was accepted. You have also received service for the function.

Assume you have now decided to install an updated version of the function. To prepare for this, you want to delete the FMID of the current function from the GLOBALZONE entry, as well as delete the service and associated HOLDDATA that were received for that function. You can use these commands:

```
SET      BDY(GLOBAL)      /* Process global zone. */.
REJECT  DFMID(HMX1101)    /* Delete FMID and reject */
        NOFMID            /* for FMIDs not in GZONE. */.
```

**Note:** This deletes SYSMODs and associated HOLDDATA for **all** functions that are not defined in the GLOBALZONE entry. It is not limited to entries for HMX1101.

To limit the REJECT command to specific functions, use the FORFMID operand in another REJECT mode.

### Example 9: Deleting Service for a Group of Source IDs

Assume you are maintaining two systems, and you want to delete from the global zone and SMPPTS all service from 1993 service levels already applied and accepted on both of your systems. The distribution zones associated with these two systems are MVS1DLIB and MVS2DLIB. You can use the following commands:

```
SET      BDY(GLOBAL)      /* Process the global zone. */.
REJECT   /* Reject from */.
          PURGE(MVS1DLIB  /* zones MVS1DLIB */.
            MVS2DLIB)    /* and MVS2DLIB */.
          PTFS           /* PTFs accepted */.
          SOURCEID(PUT93*) /* from 1993 service levels.*/.
```

### Example 10: Rejecting Selected SYSMODs That Have Been Superseded (Select Mode)

Assume you have applied but not yet accepted PTF UZ45678, which supersedes a previous PTF, UZ01234. You had received the superseded PTF, but had never installed it. For cleanup purposes, you want to reject the superseded PTF. The simplest way to do this is by specifying the BYPASS(APPLYCHECK) operand. That way, you do not need to indicate the specific zones in which the SYSMOD is superseded:

```
SET      BDY(GLOBAL)      /* Set to global zone. */.
REJECT   S(UZ01234)      /* Reject this SYSMOD, which*/.
          BYPASS(        /* was superseded by a */.
            APPCHK)     /* SYSMOD that was applied. */.
```

## Processing

REJECT processing includes these steps:

1. Selecting the eligible SYSMODs and HOLDDATA
2. Deleting the entries and related data sets for those SYSMODs

### Selecting the Eligible SYSMODs and HOLDDATA

SMP/E checks the operands specified on the REJECT command to determine which SYSMODs and HOLDDATA are eligible. First, SMP/E determines what type of REJECT processing was requested:

- If neither **SELECT**, **PURGE**, nor **NOFMID** was specified, it does mass-mode processing.
- If **SELECT** was specified, it does select-mode processing.
- If **PURGE** was specified, it does PURGE-mode processing.
- If **NOFMID** was specified, it does NOFMID-mode processing.

SMP/E selects the eligible SYSMODs and HOLDDATA according to the mode of processing.

### Mass-Mode Processing

In mass-mode processing, SMP/E selects only SYSMODs that have not been applied or accepted anywhere. First, SMP/E checks each SYSMOD to see if it meets the requirements defined by the operands specified on the REJECT command.

- If you specify the EXCLUDE operand, SMP/E makes sure the SYSMOD was not specified in the exclude list.
- If any SYSMOD types were specified (APARS, FUNCTIONS, PTFS, or USERMODS), SMP/E selects only SYSMODs that match one of the specified types. If no SYSMOD types were specified, only PTFSs are selected.
- If you specify the FORFMID operand, SMP/E makes sure that either the FMID value on one of the ++VER statements within the SYSMOD or the SYSMOD ID itself matches either an FMID specified in FORFMID or one of the FMID values contained in a specified FMIDSET.
- If you specify the SOURCEID operand, SMP/E makes sure one of the SOURCEIDs of the SYSMOD matches a source ID that you have specified, either explicitly or implicitly, on the SOURCEID operand.

Next, SMP/E determines which of these SYSMODs have not been applied or accepted anywhere. To do this, it checks all the target and distribution zones defined by zone index subentries, minus any zones or ZONESETs specified on the EXCLUDEZONE operand. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a SYSMOD is not installed in any of the zones that were checked, it may be rejected.

**Note:** A SYSMOD is considered installed if the ERROR indicator in its entry is off, or if it has been superseded. A deleted SYSMOD is not considered installed.

Each SYSMOD that meets **all** the specified conditions is eligible to be rejected.

**Note:** Superseded SYSMODs are rejected in mass mode only if the EXCLUDEZONE operand specifies the zone where the SYSMOD is superseded. EXCLUDEZONE cannot exclude all target and distribution zones from processing.

Deleted SYSMODs are rejected in mass mode provided they are eligible, regardless of whether EXCLUDEZONE is specified.

If **HOLDDATA** was specified, HOLDDATA entries associated with eligible SYSMOD entries are also eligible to be rejected. However, HOLDDATA entries that have no associated SYSMOD entries are not eligible.

If **HOLDDATA** was not specified, no HOLDDATA entries are eligible to be rejected.



### Select-Mode Processing

In select-mode processing, SMP/E selects only SYSMODs that were specified on the SELECT operand. Generally, SMP/E chooses only SYSMODs that have not been applied or accepted anywhere. To determine this, it checks all the target and distribution zones defined by zone index subentries, minus any zones or ZONESETs specified on the EXCLUDEZONE operand. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a SYSMOD is not installed in any of the zones that were checked, it may be rejected.

**Note:** A SYSMOD is considered installed if the ERROR indicator in its entry is off, or if it has been superseded. A deleted SYSMOD is not considered installed.

However, if **BYPASS** was also specified, SMP/E can select SYSMODs that have been installed.

- If **BYPASS (APPLYCHECK)** was specified, SMP/E selects the specified SYSMODs, even if they have been applied, but not if they have been accepted.
- If **BYPASS (ACCEPTCHECK)** was specified, SMP/E selects the specified SYSMODs, even if they have been accepted, but not if they have been applied.
- If **BYPASS (APPLYCHECK, ACCEPTCHECK)** was specified, SMP/E selects the specified SYSMODs, even if they have been accepted, applied, or both.

**Note:** Superseded SYSMODs are rejected in select mode only in these cases:

- The appropriate **BYPASS** operand is specified.
- The **EXCLUDEZONE** operand specifies the zones where the SYSMODs are superseded. **EXCLUDEZONE** cannot exclude all target and distribution zones from processing.

Deleted SYSMODs are rejected in select mode provided they are eligible, regardless of whether **BYPASS** or **EXCLUDEZONE** is specified.

If **HOLDDATA** was specified, **HOLDDATA** entries associated with eligible SYSMOD IDs are also eligible to be rejected, regardless of whether an associated SYSMOD entry exists.

If **HOLDDATA** was not specified, no **HOLDDATA** entries are eligible to be rejected.

### PURGE-Mode Processing

In PURGE-mode processing, SMP/E selects only SYSMODs that have been installed in the specified distribution zones and target zones to which they are applicable.

First, SMP/E checks whether a SYSMOD type, **FORFMID**, or **SOURCEID** was specified. If so, SYSMODs must meet the requirements defined by those operands:

- If you specify one or more of the SYSMOD-type operands (that is, **FUNCTIONS**, **PTFS**, **APARS**, or **USERMODS**), SMP/E checks to make sure the SYSMOD type was one of those specified.

If you do not specify a SYSMOD-type operand, the default is for SMP/E to process only PTF SYSMODs.

- If you specify the FORFMID operand, SMP/E makes sure that either the FMID value on one of the ++VER statements within the SYSMOD or the SYSMOD ID itself matches either an FMID specified in FORFMID or one of the FMID values contained in a specified FMIDSET.
- If you specify the SOURCEID operand, SMP/E makes sure one of the SOURCEIDs of the SYSMOD matches a source ID that you have specified, either explicitly or implicitly, on the SOURCEID operand.

A SYSMOD is eligible to be rejected if it meets both of these conditions:

- It is installed in at least one of the distribution zones specified on the PURGE operand.
- It is successfully installed, superseded, or deleted in all the specified distribution zones to which it is applicable.

To determine applicability, SMP/E checks the specified distribution zones. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a ZONESET was specified, SMP/E uses only the distribution zones defined in the ZONESET and ignores the target zones. If there are no distribution zones among all the zones and ZONESETs specified, REJECT processing fails. SMP/E determines whether a SYSMOD is applicable to a zone according to the type of SYSMOD.

- A base function is applicable to a zone if its SREL matches an SREL defined for the zone.
- Other types of SYSMODs are applicable to a zone if they meet either of the following sets of conditions:
  - The SREL matches an SREL defined for the zone and the SYSMOD is for an FMID that is installed in the zone.
  - The SREL matches an SREL defined for the zone and the SYSMOD is for an FMID that has been received but has not yet been installed in the zone.

If **TARGETZONE** was specified, the selected SYSMODs must also be installed in the specified target zones to which they are applicable. If a SYSMOD is not applicable to any of the specified target zones, it may still be eligible if it is installed in the specified distribution zones where it is applicable.

To determine applicability, SMP/E checks the specified target zones. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a ZONESET was specified, SMP/E uses only the target zones defined in the ZONESET and ignores the distribution zones. If there are no target zones among all the zones and ZONESETs specified, REJECT processing fails. (SMP/E determines applicability to target zones in the same way as it determines applicability to distribution zones.)

Each SYSMOD that meets all the specified conditions is eligible to be rejected.

If **HOLDDATA** was specified, HOLDDATA entries are eligible to be rejected. This includes HOLDDATA entries associated with eligible SYSMOD entries, as well as HOLDDATA entries having no associated SYSMOD entries but that meet the conditions specified by other REJECT operands.

If **FORFMID** or **SOURCEID** was specified, **HOLDDATA** entries that have no associated **SYSMOD** entries are not eligible to be rejected.

### NOFMID-Mode Processing

In **NOFMID**-mode processing, **SMP/E** selects only **SYSMODs** that are applicable to **FMIDs** not defined in the **GLOBALZONE** entry. Before selecting the eligible **SYSMODs**, **SMP/E** first checks whether **DELETEDFMID** was specified. If so, it deletes the specified **FMIDs** from the **GLOBALZONE** entry. It then compares the **SYSMOD** entries in the global zone with the **FMIDs** in the **GLOBALZONE** entry.

- A base function is eligible to be rejected if its **FMID** does not match an **FMID** in the **GLOBALZONE** entry.
- A dependent function is eligible to be rejected if its **FMID** does not match an **FMID** in the **GLOBALZONE** entry and if either of these conditions is met:
  - The **FMID** specified on the **++VER** statement does not match an **FMID** in the **GLOBALZONE** entry.
  - The **FMIDs** match, but the **SREL** specified on the **++VER** statement does not match an **SREL** in the **GLOBALZONE** entry.

If all the **++VER** statements in the function meet these conditions, the function is eligible to be rejected.

- For other types of **SYSMODs**, **SMP/E** checks whether either of these conditions is met:
  - The **FMID** specified on the **++VER** statement does not match an **FMID** in the **GLOBALZONE** entry.
  - The **FMIDs** match, but the **SREL** specified on the **++VER** statement does not match an **SREL** in the **GLOBALZONE** entry.

If all the **++VER** statements in the **SYSMOD** meet these conditions, the **SYSMOD** is eligible to be rejected.

If **HOLDDATA** was specified, **HOLDDATA** entries associated with eligible **SYSMOD** entries are also eligible to be rejected. In addition, **HOLDDATA** entries whose associated **FMID** is not defined in the global zone are eligible for rejection.

If **HOLDDATA** was not specified, only **HOLDDATA** entries associated with eligible **SYSMOD** entries are eligible to be rejected. However, **HOLDDATA** entries whose associated **FMID** is not defined in the global zone are not eligible.

## Processing the **SYSMODs** and **HOLDDATA**

Once **SMP/E** has selected all the eligible **SYSMODs** and **HOLDDATA**, it rejects the appropriate entries and associated **SMPTLIB** data sets. For each eligible **SYSMOD**, **SMP/E** deletes the following:

- The **SMPPTS MCS** entry.
- The global zone **SYSMOD** entry.
- The associated **FMID** subentry in the **GLOBALZONE** entry.

The **FMID** subentry is deleted for a function **SYSMOD** if both of these conditions are met:

- Mass-mode or select-mode processing was done.

- **BYPASS** was not specified.

**Note:** Once the FMID subentry is deleted, service for that function is no longer received. If you intend to install a more recent copy of the function, you can use UCLIN to add the FMID subentry back to the GLOBALZONE entry. This way, no service is lost in the meantime.

- The eligible HOLDDATA entries.
- The associated SMPTLIB data sets, if the SYSMOD was packaged in RELFILE format. SMP/E determines the number of data sets to delete from the FILES operand of the header MCS.

SMP/E locates the SMPTLIB data sets to be deleted by checking the following sources in the order shown. If SMP/E finds one of the data sets, it assumes that all of the SMPTLIB data sets can be found the same way.

1. If the SMPTLIB data sets are cataloged, they are deleted according to the catalog.
2. If there is an SMPTLIB DD statement, the data sets are deleted from the volumes specified on the DD statement.
3. If there is an SMPTLIB DDDEF entry, the data sets are deleted from the volumes specified in the DDDEF entry.

If the SMPTLIB data sets are not located by the catalog and there is no DD statement or DDDEF entry for them, the SYSMOD is not rejected. If there is a DD statement or DDDEF entry but the data sets are not found, SMP/E issues a warning message and continues REJECT processing for that SYSMOD.

**Note:** If any elements were packaged in either LKLIB or TXLIB format, no action is taken on those data sets.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of REJECT processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone — Read without enqueue.

### 2. REJECT processing

Global zone — Update with exclusive enqueue.

SMPPTS — Update with exclusive enqueue.

DLIB zone — Read with shared enqueue.

Target zone — Read with shared enqueue.

### Notes:

- a. The distribution zones are accessed in mass mode, select mode, and PURGE mode.
- b. The target zones are accessed in mass mode and select mode, and in PURGE mode if the TARGETZONE operand was specified.

3. Termination

All resources are freed.





---

## Chapter 14. The REPORT CALLLIBS Command

This command helps you to identify and relink load modules when implicitly-included modules in a particular library are updated. Specifically, the REPORT CALLLIBS command provides a report of those load modules that have a CALLLIBS subentry list in their LMOD entry (and therefore use the automatic library call option to implicitly include modules from a specified library). If requested, REPORT CALLLIBS also creates jobs to relink the load modules that are reported on. A separate job is created for each affected zone.

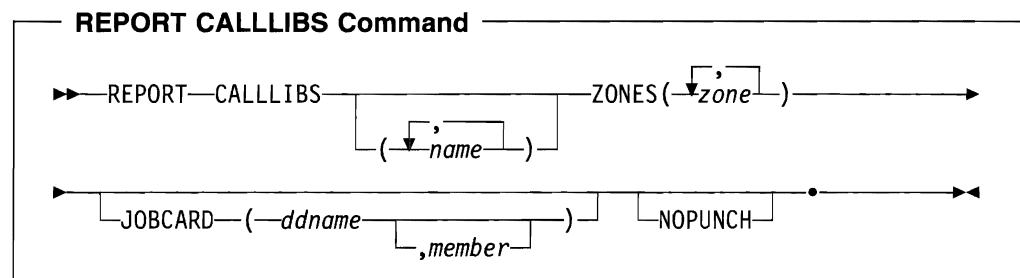
---

### Zones for SET BOUNDARY

For the REPORT CALLLIBS command, the SET BOUNDARY command must specify the global zone.

---

### Syntax



### Operands

#### CALLLIBS

requests a report about load modules whose LMOD entries contain a CALLLIBS subentry list. The specified name is a DDDEF named in an LMOD entry's CALLLIBS subentry list.

If you want to report on only those LMOD entries with particular CALLLIBS subentries, specify CALLLIBS and the specific names. If you want to report on all LMOD entries with CALLLIBS subentries, specify CALLLIBS without any names.

CALLLIBS is a required operand for the REPORT CALLLIBS command.

**Note:** CALLLIBS is mutually exclusive with CROSSZONE, ERRSYSMODS, SOURCEID, and SYSMODS.

#### JOBCARD

indicates where SMP/E is to get the job card for the jobs punched to the SMPPUNCH data set. The *ddname* is for the partitioned data set containing the job card member. *member* specifies the name of the member within the data set containing the job card.

### Notes:

1. JOBCARD is allowed only on the REPORT CALLLIBS command.
2. When target zones are being reported on and NOPUNCH is **not** specified, JOBCARD is a required operand.  
If NOPUNCH is specified, or all the zones being processed are DLIB zones, JOBCARD is ignored.
3. If *member* is not specified, the default is JOBCARD.

### NOPUNCH

indicates that SMP/E should not write any output to SMPPUNCH. If **NOPUNCH** is **not** specified, JCL or commands are written to SMPPUNCH. This output contains JCL or commands that can be used for further processing.

**Note:** The output produced by REPORT CALLLIBS processing contains JCL that can be used to relink load modules that contain a CALLLIBS sub-entry list. JCL is **not** written to SMPPUNCH for any distribution zones specified in the ZONES list.

### ZONES

specifies which zones SMP/E should report on. This list can include zone names, ZONESET names, or both. You can specify the target zones, distribution zones, or any combination of these. SMP/E produces a separate report for each zone it checks.

For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. For example, suppose you have a ZONESET named MVS1 and a zone named MVS1. If you specify **MVS1** on this operand, SMP/E assumes you want to use the zones defined in ZONESET MVS1 (which might or might not include zone MVS1), and not the individual zone MVS1.

ZONES is a required operand on the REPORT CALLLIBS command.

---

## Data Sets Used

The following data sets may be needed to run the REPORT CALLLIBS command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPRPT	<i>jobcard</i>
SMP_CSI	SMPOUT	SMPSNAP	<i>zone</i>
SMPLOG	SMPPUNCH		

### Notes:

1. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.
2. *jobcard* represents the ddname specified on the JOBCARD operand of the REPORT CALLLIBS command. It is required if SMPPUNCH output is to be generated by the REPORT CALLLIBS command.



3. The REPORT CALLLIBS command may require that DDDEF entries needed to generate SMPPUNCH output be available in the zone being processed. To generate the SMPPUNCH output, the REPORT CALLLIBS command needs the following:

- A DDDEF for the SMPLTS data set
- Any DDDEFs named in a selected LMOD entry's CALLLIBS subentry list
- Any DDDEFs named in a selected LMOD entry's SYSLIB subentry list
- A DDDEF or an entry in GIMMPDFT for SYSUT1

The data sets themselves are not required to run the REPORT CALLLIBS command.

---

## Output

Output from the REPORT CALLLIBS command includes reports, as well as data written to SMPPUNCH.

## Reports

The following reports are produced during REPORT CALLLIBS processing:

- CALLLIBS Summary report
- File Allocation report

See Chapter 31 for descriptions of these reports.

## SMPPUNCH Output

To make it easier for you to relink load modules whose LMOD entries contain a particular CALLLIBS dname, SMP/E writes JCL to the SMPPUNCH data set. JCL is not written to SMPPUNCH in the following cases:

- The zone being processed is a DLIB zone.
- **NOPUNCH** was specified on the REPORT CALLLIBS command.

Figure 6 on page 278 shows the format of the SMPPUNCH output from the REPORT CALLLIBS command.

## REPORT CALLLIBS Command

```
//zone      JOB ....
//*SMPLTS
//*****
//*
//* THE FOLLOWING JOB WAS GENERATED FROM THE REPORT CALLLIBS COMMAND
//* FOR TARGET ZONE zone      ON yy.ddd AT hh:mm:ss
//*
//*****
//LINKxxxx EXEC PGM=link-edit utility name,           EXECLNK
//      PARM=(lparm1, lparm2, . . ., lparmx, CALL)
//*****
//*
//* LMODS LINKED IN THIS STEP:
//*
//*      aaaaaaaa   bbbbbbbb   cccccccc   dddddddd   eeeeeeee
//*      ffffffff   gggggggg   hhhhhhhh   iiiiii    jjjjjjjj
//*
//*
//*****
//SYSLIB  DD DSN=calllibs dsname,DISP=SHR              SSSSSSS
//      DD DSN=calllibs dsname,DISP=SHR              SSSSSSS
//SYSPRINT DD information from SYSPRINT DDDEF or SYSPRINT entry  SSSSSSS
//      in GIMMPDFT table or SMP/E SYSPRINT default SYSOUT=*    SSSSSSS
//SYSLMOD DD information from LMOD's SYSLIB DDDEF        SSSSSSS
//SYSUT1  DD SYSUT1 DDDEF or SYSUT1 entry in GIMMPDFT table  SSSSSSS
//SMPLTS  DD information from SMPLTS DDDEF              SSSSSSS
//SYSLIN  DD *
link-edit control cards for lmod aaaaaaaa
INCLUDE SMPLTS(aaaaaaaa)
NAME aaaaaaaa(R)
.
.
.
link-edit control cards for lmod jjjjjjjj
INCLUDE SMPLTS(jjjjjjjj)
NAME jjjjjjjj(R)
/*
```

Figure 6 (Part 1 of 2). REPORT CALLLIBS: Format of SMPPUNCH Output

```

//LINKxxxx EXEC PGM=IEWBLINK,
//      PARM=(
//      OPTIONS(GENOPTS)
//      )
//*****
//*
//*  LMODS LINKED IN THIS STEP:
//*
//*      kkkkkkkk   11111111   mmmmmmmmm   nnnnnnnn   00000000
//*
//*
//*****
//SYSLIB  DD DSN=calllibs dsname from DDDEF,DISP=SHR          SSSSSSSS
//      DD DSN=calllibs dsname from DDDEF,DISP=SHR          SSSSSSSS
//SYSPRINT DD information from SYSPRINT DDDEF or SYSPRINT entry  SSSSSSSS
//      in GIMMPDFT table or SMP/E SYSOUT default          SSSSSSSS
//SYSLMOD DD information from LMOD's SYSLIB DDDEF            SSSSSSSS
//SYSUT1  DD SYSUT1 DDDEF or SYSTU1 entry in GIMMPDFT table  SSSSSSSS
//SMPLTS  DD information from SMPLTS DDDEF                  SSSSSSSS
//GENOPTS DD *
link-edit parameters . . . CALL
//SYSLIN  DD *
link-edit control cards for lmod kkkkkkkk
INCLUDE SMPLTS(kkkkkkkk)
NAME kkkkkkkk(R)
      |
      |
link-edit control cards for lmod 00000000
INCLUDE SMPLTS(00000000)
NAME 00000000(R)
/*

```

Figure 6 (Part 2 of 2). REPORT CALLLIBS: Format of SMPPUNCH Output

## Example: Using REPORT CALLLIBS

Assume you have installed service into libraries CSSLIB, PLIBASE, and HFSCLIB1. Your system might contain load modules that implicitly include routines from these libraries, and you want to make sure the affected load modules include the latest routines from the upgraded libraries. ZONESET entry TZONSET identifies the target zones associated with the affected load modules. To identify the load modules that might need to be updated, use the following REPORT CALLLIBS command:

```

* REPORT CALLLIBS(CSSLIB,PLIBASE,HFSCLIB1) /* Report on ddnames */
      ZONES(TZONSET) /* for ZONESET TZONSET */
      JOBCARD(JDDNAME) /* for JOBCARD JDDNAME. */.

```

Figure 7 on page 280 shows an example of the report SMP/E produces. Each part of the figure shows the report for a separate zone in the ZONESET.

## REPORT CALLLIBS Command

```
PAGE nnnn - NOW SET TO GLOBAL ZONE          DATE mm/dd/yyTIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

                                CALLLIBS SUMMARY REPORT FOR TARGET ZONE MVSTZ1

CALLLIBS  LMOD      LMOD      STEPNAME IN  CALLLIBS  DATA SET NAME
DDNAME    NAME      SYSLIB     JOB MVSTZ1  ALLOCATION  OR PATH

CSSLIB    IEELoad1  LINKLIB   LINK0001    CSSLIB    SYS1.CSSLIB
          IEFLOAD5  LINKLIB   LINK0001    CSSLIB    SYS1.CSSLIB
          IGGLOAD3  LPALIB    LINK0002    CSSLIB    SYS1.CSSLIB
          PLIBASE   SYS1.PLIBASE
          IWWLOAD6  LPALIB    LINK0002    CSSLIB    SYS1.CSSLIB
          PLIBASE   SYS1.PLIBASE

HFSCLIB1  BPXLMOD4  BPXLIB1   LINK0003    HFSCLIB1  '/this/pathname/fits/on/one/record/in/the/report/'

PLIBASE   IGGLOAD3  LPALIB    LINK0002    CSSLIB    SYS1.CSSLIB
          PLIBASE   SYS1.PLIBASE
          IWWLOAD6  LPALIB    LINK0002    CSSLIB    SYS1.CSSLIB
          PLIBASE   SYS1.PLIBASE
```

Figure 7 (Part 1 of 2). Example of a CALLLIBS Summary Report

```
PAGE nnnn - NOW SET TO GLOBAL ZONE          DATE mm/dd/yyTIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

                                CALLLIBS SUMMARY REPORT FOR TARGET ZONE MVSTZ2

CALLLIBS  LMOD      LMOD      STEPNAME IN  CALLLIBS  DATA SET NAME
DDNAME    NAME      SYSLIB     JOB         ALLOCATION  OR PATH

***NONE
```

Figure 7 (Part 2 of 2). Example of a CALLLIBS Summary Report

Because **NOPUNCH** was not specified, SMP/E also writes the JCL shown in Figure 8 on page 281 to the SMPPUNCH data set. Use the STEPNAME IN JOB column of the CALLLIBS Summary report to find the associated job step in the JCL.

```

//MVSTZ1 JOB ....
//*SMPLTS
//*****
//*
//* THE FOLLOWING JOB WAS GENERATED FROM THE REPORT CALLLIBS COMMAND
//* FOR TARGET ZONE MVSTZ1 ON yy.ddd AT hh:mm:ss
//*
//*****
//LINK001 EXEC PGM=IEWBLINK,                               EXECLNK
//      PARM=('RENT',
//      'XREF,LIST,LET,CALL')
//*****
//*
//* LMODS LINKED IN THIS STEP:
//*
//*      IEELoad1      IEFLOAD5
//*
//*****
//SYSPRINT DD SYSOUT=*                                     DEFAULT
//SMPLTS   DD DSN=SYS1.MVSTZ1.SMPLTS,                     SMPLTS
//          DISP=(OLD)                                    SMPLTS
//SYSLMOD  DD DSN=SYS1.LINKLIB,                             LINKLIB
//          DISP=(OLD)                                    LINKLIB
//SYSUT1   DD UNIT=SYSALLDA,                               SYSUT1
//          DISP=(NEW,DELETE),                            SYSUT1
//          SPACE=(3120,(380,760))                        SYSUT1
//SYSLIB   DD DSN=SYS1.CSSLIB,                             CSSLIB
//          DISP=(OLD)                                    CSSLIB
//SYSLIN   DD *                                           DEFAULT
ORDER IEE001
ORDER IEE002
ORDER IEE003
ALIAS GETRTN
ENTRY IEE001
INCLUDE SMPLTS(IEELoad1)
NAME IEELoad1(R)
ORDER IEF001
ORDER IEF002
ENTRY IEF001
INCLUDE SMPLTS(IEFLOAD5)
NAME IEFLOAD5(R)
/*

```

Figure 8 (Part 1 of 3). Example of SMPPUNCH Output for REPORT CALLLIBS

## REPORT CALLLIBS Command

```
//LINK0002 EXEC PGM=IEWBLINK,                               EXECLNK
//      PARM=('REUS',
//      'XREF,LIST,LET,CALL')
//*****
//*
//*  LMODS LINKED IN THIS STEP:
//*
//*      IGGLOAD3      IWWLOAD6
//*
//*****
//SYSPRINT DD SYSOUT=*                                     DEFAULT
//SMPLTS   DD DSN=SYS1.MVSTZ1.SMPLTS,                      SMPLTS
//          DISP=(OLD)                                     SMPLTS
//SYSLMOD  DD DSN=SYS1.LPALIB,                              LPALIB
//          DISP=(OLD)                                     LPALIB
//SYSUT1   DD UNIT=SYSALLDA,                                SYSUT1
//          DISP=(NEW,DELETE),                             SYSUT1
//          SPACE=(3120,(380,760))                         SYSUT1
//SYSLIB   DD DSN=SYS1.CSSLIB,                              CSSLIB
//          DISP=(OLD)                                     CSSLIB
//          DD DSN=SYS1.PLIBASE,                            PLIBASE
//          DISP=(OLD)                                     PLIBASE
//SYSLIN   DD *                                           DEFAULT
ORDER IGG001
ALIAS BMHELP
ALIAS CMHELP
ENTRY IGG001
INCLUDE SMPLTS(IGGLOAD3)
NAME IGGLOAD3(R)
ORDER IWW001
ORDER IWW002
ENTRY IWW001
INCLUDE SMPLTS(IWWLOAD6)
NAME IWWLOAD6(R)
/*
```

Figure 8 (Part 2 of 3). Example of SMPPUNCH Output for REPORT CALLLIBS

```

//LINK0003 EXEC PGM=IEWBLINK,                                EXECLNK
//          PARM=('RENT',
//          'XREF,LIST,LET,CASE(MIXED),CALL')
//*****
//*
//*  LMODS LINKED IN THIS STEP:
//*
//*    BPXLMOD4
//*
//*****
//SYSPRINT DD SYSOUT=*                                       DEFAULT
//SMPLTS  DD DSN=SYS1.MVSTZ1.SMPLTS,                         SMPLTS
//          DISP=(OLD)                                       SMPLTS
//SYSLMOD  DD PATH='/bin/etc/IBM/'                            BPXLIB1
//*LIBRARYDD=BPXLIB1                                         BPXLIB1
//SYSUT1   DD UNIT=SYSALLDA,                                  SYSUT1
//          DISP=(NEW,DELETE),                               SYSUT1
//          SPACE=(3120,(380,760))                           SYSUT1
//SYSLIB   DD PATH='/this/pathname/fits/on/one/record/in/the/report/' HFSCLIB1
//*LIBRARYDD=HFSCLIB1                                       HFSCLIB1
//SYSLIN   DD *                                              DEFAULT
//          ALIAS './friendly/name4'
//          ENTRY BPXMOD1
//          INCLUDE SMPLTS(BPXLMOD4)
//          NAME BPXLMOD4(R)
//          /*

```

Figure 8 (Part 3 of 3). Example of SMPPUNCH Output for REPORT CALLLIBS

## Processing

The REPORT CALLLIBS command checks the specified zones for load modules whose LMOD entries contain a CALLLIBS subentry list with the specified dnames. If requested, it also writes JCL to the SMPPUNCH data set so that these load modules can be relinked.

SMP/E first verifies that each target and distribution zone specified on the ZONES operand is defined in the global zone. The zones and ZONESETs specified on the ZONES operand are used to create a list of zones to be reported on. If the global zone is specified either separately or as a part of a ZONESET, SMP/E ignores it and continues processing. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a zone is specified separately and is also part of a ZONESET, it is only reported on once.

Next, SMP/E opens the zones for read access. For each of the zones being processed, SMP/E verifies that the zone type in the ZONEINDEX subentry matches the zone type in the zone definition entry.

SMP/E then reads through all the zones being processed by the REPORT CALLLIBS command. If a specific CALLLIBS name has been specified on the REPORT CALLLIBS command, SMP/E processes only those LMOD entries that contain CALLLIBS subentries with that name. If no CALLLIBS names have been

specified, SMP/E processes all LMOD entries containing CALLLIBS subentries. A CALLLIBS Summary report is generated for each zone that is processed, and the affected LMOD entries are listed.

If **any** of the following occurs, SMP/E issues an error message, and REPORT CALLLIBS processing stops:

- A specified zone or ZONESET does not exist in the global zone.
- A zone within a specified ZONESET does not exist in the global zone.
- Open or enqueue processing fails for a zone.
- A zone definition entry could not be found for the specified zone.
- The zone type in the ZONEINDEX subentry does not match the zone type in the zone definition entry.

In addition, if **NOPUNCH** is **not** specified (and the zone being processed is a target zone), SMP/E writes JCL to the SMPPUNCH data set. This JCL can be used to link-edit the identified load modules. If **NOPUNCH** is specified, SMP/E does not write any output to SMPPUNCH.

The source and format of the JCL punched by the REPORT CALLLIBS command is similar to that of the GENERATE command. Table 18 shows the source used to create JCL for the REPORT CALLLIBS command.

<i>Table 18 (Page 1 of 2). Sources of Information for REPORT CALLLIBS Output JCL</i>	
<b>Source of Input</b>	<b>JCL It Is Used For</b>
Zone name	Job name
<ul style="list-style-type: none"> <li>• The library and member specified on the JOBCARD operand of the REPORT CALLLIBS command. If a member name is not specified, JOBCARD is used as the default member name.</li> <li>• The ddname specified on the JOBCARD operand refers to a DD statement or a DDDEF entry in the global zone. If neither is found, or the JOBCARD operand was not specified at all, SMP/E issues an error message and generates a comment card for the job card.</li> </ul>	Job card
One of the following: <ul style="list-style-type: none"> <li>• Link-edit utility program name from OPTIONS entry defined for the zone</li> <li>• SMP/E default link-edit utility program name</li> </ul>	Link-edit utility on EXEC statement
Link-edit utility parameters from the LMOD entry plus CALL and either of the following: <ul style="list-style-type: none"> <li>• Link-edit utility parameters from the OPTIONS entry defined for the current zone. or</li> <li>• SMP/E default link-edit parameters</li> </ul>	Link-edit parameters
DDDEF for LMOD entry's SYSLIB from zone being processed	SYSLMOD DD statement



<i>Table 18 (Page 2 of 2). Sources of Information for REPORT CALLLIBS Output JCL</i>	
<b>Source of Input</b>	<b>JCL It Is Used For</b>
One of the following: <ul style="list-style-type: none"> <li>• DDDEF named in link-edit UTILITY entry from OPTIONS entry defined for zone</li> <li>• DDDEF for SYSPRINT from zone being processed</li> <li>• SYSPRINT entry in GIMMPDFT</li> <li>• SMP/E default (SYSOUT=*)</li> </ul>	SYSPRINT DD statement
One of the following: <ul style="list-style-type: none"> <li>• DDDEF for SYSUT1 from zone being processed</li> <li>• SYSUT1 entry in GIMMPDFT</li> </ul>	SYSUT1 DD statement
DDDEFs named in LMOD entry's CALLLIBS subentries	SYSLIB DD statement
DDDEF for SMPLTS in zone being processed	SMPLTS DD statement
Link-edit control statements from LMOD entry SMP/E generates the following: <ul style="list-style-type: none"> <li>• INCLUDE statement for LMOD entry from SMPLTS data set</li> <li>• NAME statement with REPLACE option for LMOD entry</li> </ul>	Link-edit control statement

Load modules with the same SYSLMOD libraries, link-edit attributes, and SYSLIB (CALLLIBS) concatenations are generally linked in the same step. SMP/E generates a unique step (LINKxxxx) within each job. The CALLLIBS Summary report cross-references the LMOD entries with the steps in which they are linked.

The REPORT CALLLIBS command processes the link-edit parameters of the load modules in SMPPUNCH output as follows:

- If the parameter string to the link-edit utility for a load module does not exceed 100 characters, the parameters are written on the PARM operand of the EXEC statement.
- If the parameter string exceeds 100 characters, and the DFP level of the driving system on which SMP/E is running supports the binder, SMP/E specifies only the OPTIONS option on the EXEC statement. GENOPTS is specified as the ddname in the OPTIONS option, and a DD statement is created for GENOPTS. SMP/E adds all other options after the GENOPTS DD statement.
- If the parameter string exceeds 100 characters, and the DFP level of the driving system on which SMP/E is running does not support the binder, the parameters are truncated and SMP/E issues an error message.

**Notes:**

1. The REPORT CALLLIBS command does not write INCLUDE statements for all the modules contained in a load module. Instead, the load module is included from the SMPLTS data set. The REPLACE option is automatically written on the NAME statement.
2. The JCL generated by the REPORT CALLLIBS command is not intended to be

used as input to JCLIN processing, because it would result in misleading information being added to target or distribution zone.

---

### Zone and Data Set Sharing Considerations

The following identifies the phases of REPORT CALLLIBS processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

#### 1. Initialization

- Global zone – Read without enqueue.
- Target zones (as required) – Read without enqueue.
- DLIB zones (as required) – Read without enqueue.

#### 2. Processing

- Global zone – Read with shared enqueue.
- Target zones (as required) – Read with shared enqueue.
- DLIB zones (as required) – Read with shared enqueue.

#### 3. Termination

All resources are freed.

---

## Chapter 15. The REPORT CROSSZONE Command

This command helps you synchronize the service levels of different products when the products are installed in different zones controlled by the same global zone. Specifically, REPORT CROSSZONE lists conditional requisites that must be installed in certain zones because of SYSMODs that are installed in other zones. Information about these requisites is provided in the Cross-Zone Requisite SYSMOD report. The commands needed to install the requisites are written to the SMPPUNCH data set.

For example, suppose you have two versions of a product: one for MVS/ESA\* and one for MVS/XA. Each version is installed on a different system and in different target and distribution zones. To keep the two versions synchronized, the service for one version may specify service for the other version as a conditional requisite. However, because the versions are not in the same target or distribution zones, neither zone contains information about conditional requisites defined in the other zone. Therefore, SMP/E cannot use the information to keep the two versions synchronized. Instead, you can use the REPORT CROSSZONE command to obtain a summary of the requisite information and have SMP/E generate the commands needed to install the requisites into the appropriate zones.

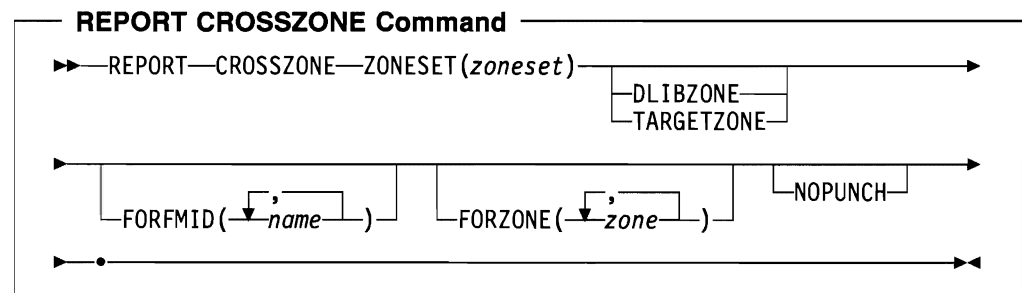
---

### Zones for SET BOUNDARY

For the REPORT CROSSZONE command, the SET BOUNDARY command must specify the global zone.

---

### Syntax



### Operands

#### CROSSZONE

requests a report about cross-zone requisites.

CROSSZONE is a required operand for the REPORT CROSSZONE command.

**Note:** CROSSZONE is mutually exclusive with CALLLIBS, ERRSYSMODS, SOURCEID, and SYSMODS.

#### DLIBZONE

indicates that SMP/E should report only on SYSMODs accepted into the distribution zones in the ZONESET.

DLIBZONE is required if the ZONESET being used contains both target and distribution zones, because the REPORT CROSSZONE command processes only zones of the same type. It is not required to process a ZONESET containing only distribution zones.

**Notes:**

1. DLIBZONE is allowed only on the REPORT CROSSZONE command.
2. DLIBZONE is mutually exclusive with TARGETZONE.
3. **DLIBZONE** can also be specified as **DZONE**.

**FORFMID**

specifies the FMIDs used to limit which SYSMODs are included in the report. This list can include FMIDs, FMIDSET names, or both.

If **FORFMID** is specified, SMP/E lists only the SYSMODs that are needed for these FMIDs.

If **FORFMID** is not specified, SMP/E reports on requisites based on the CIFREQ subentries in the SYSMOD entries for **all** the functions in the ZONESET zones.

**Note:** CIFREQ subentries list requisites for the function that were specified on ++IF statements in other SYSMODs. They also list the causer SYSMOD that contained the ++IF statement. For more information about CIFREQ subentries and conditional requisites, see “Conditional Requisites (IFREQ)” on page 32.

**FORZONE**

specifies the zones to be reported on. SMP/E lists only the SYSMODs that are needed in these zones. All these zones must be part of the ZONESET being used.

If **FORZONE** is not specified, SMP/E reports on requisites for all the zones in the ZONESET.

**Note:** FORZONE is allowed only on the REPORT CROSSZONE command.

**NOPUNCH**

indicates that SMP/E should not write any output to SMPPUNCH. If **NOPUNCH** is **not** specified, JCL or commands are written to SMPPUNCH. This output contains JCL or commands that can be used for further processing.

**Note:** The output produced by REPORT CROSSZONE processing contains commands for installing cross-zone requisites.

**TARGETZONE**

indicates that SMP/E should report only on SYSMODs applied to the target zones in the ZONESET. **TARGETZONE** is required if the ZONESET being used contains both target and distribution zones, because the REPORT CROSSZONE command processes only zones of the same type. It is not required to process a ZONESET containing only target zones.

**Notes:**

1. **TARGETZONE** is allowed only on the REPORT CROSSZONE command.
2. **TARGETZONE** is mutually exclusive with **DLIBZONE**.
3. **TARGETZONE** can also be specified as **TZONE**.

### ZONESET

is the name of the global zone ZONESET entry that is used to report on cross-zone requisites. SMP/E checks all the zones in the ZONESET for information on conditional requisites that have been installed.

ZONESET is a required operand for the REPORT CROSSZONE command.

For more information about defining a ZONESET, see "ZONESET Entry (Global Zone)" on page 794.

---

## Data Sets Used

The following data sets may be needed to run the REPORT CROSSZONE command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMPCNTL	SMPLOGA	SMPPUNCH	SMPSNAP
SMPCSI	SMPOUT	SMPRPT	<i>zone</i>
SMPLPG			

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

---

## Usage Notes

If you use the REPORT CROSSZONE command, keep these considerations in mind:

- After installing the requisite SYSMODs identified by the REPORT CROSSZONE command, you should run the command again to see whether that installation causes any new requisites. You should repeat this process until there are no new requisites.
- You should always check the Cross-Zone Requisite SYSMOD report for unreceived requisites before using the SMPPUNCH output from REPORT CROSSZONE. You may want to receive these SYSMODs in order to run the commands in SMPPUNCH, or you may prefer to delete them from SMPPUNCH and receive them later.

---

## Output

Output from the REPORT CROSSZONE command includes reports, as well as data written to SMPPUNCH.

## Reports

The following reports are produced during REPORT CROSSZONE processing:

- Cross-Zone Requisite SYSMOD report
- File Allocation report

See Chapter 31 for descriptions of these reports.

## SMPPUNCH Output

To make it easier for you to install cross-zone requisite SYSMODs, SMP/E writes the necessary commands to the SMPPUNCH data set: SET BOUNDARY, RESETRC, and either ACCEPT (for distribution zones) or APPLY (for target zones). Nothing is written to SMPPUNCH for a specified zone in the following cases:

- There are no requisite SYSMODs for the specified zone.
- **NOPUNCH** was specified on the REPORT CROSSZONE command.

Figure 9 shows the format of the SMPPUNCH output from the REPORT CROSSZONE command.

```
SET BDY (zone1 ).
RESETRC.
command SELECT(
           sysmod1 /* REQUIRED DUE TO sysmod2 IN zone2 */
           )
GROUP.
```

Figure 9. REPORT CROSSZONE: Format of SMPPUNCH Output

*zone1*

is the name of the zone where the requisites are to be installed.

*command*

is the command to be used to install the requisites: ACCEPT for a distribution zone, and APPLY for a target zone.

*sysmod1*

is the ID of a requisite SYSMOD.

*sysmod2*

is the ID of the SYSMOD that contained the CIFREQ data (the causer SYSMOD).

*zone2*

is the name of the zone that contained the CIFREQ data (the causer zone).

### Notes:

1. If there is more than one causer SYSMOD or causer zone for a selected SYSMOD, a comment is written for each of the causers. These comments refer to the previous SYSMOD in the select list.
2. If there are requisites for more than one zone, a set of commands is written for each zone.
3. You can edit the SMPPUNCH output before using it. For example, you may want to install SYSMODs for only one of the zones, or you may want to delete SYSMODs that have not yet been received.

## Examples

The following examples are provided to help you use the REPORT CROSSZONE command:

- “Example 1: Using REPORT CROSSZONE”
- “Example 2: Using REPORT CROSSZONE with Zones Controlled by Different Global Zones” on page 294

### Example 1: Using REPORT CROSSZONE

Assume that you have a system that supports MVS/ESA and MVS/XA. There is one target zone (MVSESA) for base MVS/ESA functions and another target zone (PRODESA) for a dependent function. Likewise, there is a target zone (MVSXA) for base MVS/XA functions and another target zone (PRODXA) for a dependent function. Table 19 shows some of the functions and PTFs contained in each zone.

*Table 19. REPORT CROSSZONE  
Example: SYSMOD Installed in Each Zone*

Zone	Functions	PTFs
MVSESA	HBB3310	UZ00005 UZ00009 UZ00011 UZ00013 UZ00031 UZ00032
PRODESA	HJS3311	UZ00006 UZ00022 UZ00025 UZ00026
MVSXA	HBB2102 JBB2220	UZ00030 UZ00031
PRODXA	HJS2220	UZ00032 UZ00033

You have also received the following SYSMODs but have not yet applied or accepted them:

UZ00023  
UZ00024  
UZ00027

Assume some of the PTFs specify conditional requisites. Table 20 on page 292 shows some of the statements in these PTFs (causer SYSMODs), along with the zones where they were installed (causer zones), the functions they are applicable to (causer FMIDs), the functions specified on the ++IF statements (IFREQ FMIDs), the zones where these functions are installed (IFREQ zones), and the requisites.

*Table 20. REPORT CROSSZONE Example: Required SYSMODs*

<b>Causer SYSMODs</b>	<b>Causer Zones and FMIDs</b>	<b>IFREQ Zones and FMIDs</b>	<b>Required SYSMODs</b>
++PTF(UZ00011). ++VER (Z038) FMID(HBB3310). ++IF FMID(HJS3311) REQ(UZ00023).	MVSESA – HBB3310	PRODESA – HJS3311	UZ00023
++PTF(UZ00013). ++VER (Z038) FMID(HBB3310). ++IF FMID(HJS3311) REQ(UZ00024).	MVSESA – HBB3310	PRODESA – HJS3311	UZ00024
++PTF(UZ00006). ++VER (Z038) FMID(HJS3311). ++IF FMID(HBB3310) REQ(UZ00009).	PRODESA – HJS3311	MVSESA – HBB3310	UZ00009
++PTF(UZ00022). ++VER (Z038) FMID(HJS3311). ++IF FMID(HBB3310) REQ(UZ00005).	PRODESA – HJS3311	MVSESA – HBB3310	UZ00005
++PTF(UZ00025). ++VER (Z038) FMID(HJS3311). ++IF FMID(HJS2220) REQ(UZ00027).	PRODESA – HJS3311	PRODXA – HJS2220	UZ00027
++PTF(UZ00026). ++VER (Z038) FMID(HJS3311). ++IF FMID(HJS2220) REQ(UZ00028).	PRODESA – HJS3311	PRODXA – HJS2220	UZ00028
++PTF(UZ00030). ++VER (Z038) FMID(HBB2102). ++IF FMID(HJS2220) REQ(UZ00032).	MVSXA – HBB2102	PRODXA – HJS2220	UZ00032
++PTF(UZ00031). ++VER (Z038) FMID(HBB2102). ++IF FMID(HJS2220) REQ(UZ00033).	MVSXA – HBB2102	PRODXA – HJS2220	UZ00033



The dependent functions are different versions of the same product. They must be synchronized with each other and with their base functions. You can set up two ZONESETs (SESA and SXA) to help keep these products at the same service level. Table 21 shows the zones contained in each ZONESET:

ZONESET	Zones
SESA	MVSESA PRODESA PRODXA
SXA	MVSXA PRODXA PRODESA

Assume you want to find out whether there are any cross-zone requisites for the zones in ZONESET SESA. You can use the following commands:

```
SET      BDY(GLOBAL)      /* Process global zone.  */.
REPORT  CROSSZONE        /* Report on requisites  */.
        ZONESET(SESA)    /* for ZONESET SESA.    */.
```

SMP/E checks zones MVSESA, PRODESA, and PRODXA because they are the zones defined in ZONESET SESA. Because FORFMID was not specified, SMP/E checks the SYSMOD entries for **all** the functions installed in those zones. It makes a list of the CIFREQ subentries for all the functions. Then, because **FORZONE** was not specified, SMP/E reports on requisites needed in **all** the zones in the ZONESET.

Figure 10 shows an example of the report SMP/E produces:

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
```

CROSSZONE REQUISITE SYSMOD REPORT FOR APPLY						
ZONE	REQUIRES		CAUSER			
NAME	FMID	SYSMOD	RECEIVED	SYSMOD	FMID	ZONE
MVSESA		NONE				
PRODXA	HJS2220	UZ00027	YES	UZ00025	HJS3311	PRODESA
	HJS2220	UZ00028	NO	UZ00026	HJS3311	PRODESA
PRODESA	HJS3311	UZ00023	YES	UZ00011	HBB3310	MVSESA
	HJS3311	UZ00024	YES	UZ00013	HBB3310	MVSESA

Figure 10. Example of a Cross-Zone Requisite SYSMOD Report

SMP/E also writes the commands shown in Figure 11 to the SMPPUNCH data set:

```
SET BDY (PRODXA ).
RESETRC.
APPLY  SELECT(
           UZ00027  /* REQUIRED DUE TO UZ00025 IN PRODESA */
           UZ00028  /* REQUIRED DUE TO UZ00026 IN PRODESA */
        )
      GROUP.
SET BDY (PRODESA).
RESETRC.
APPLY  SELECT(
           UZ00023  /* REQUIRED DUE TO UZ00011 IN MVSESA */
           UZ00024  /* REQUIRED DUE TO UZ00013 IN MVSESA */
        )
      GROUP.
```

Figure 11. Example of SMPPUNCH Output for REPORT CROSSZONE

After getting the Cross-Zone Requisite SYSMOD report, you can do the following:

1. Receive SYSMOD UZ00028 so you can install it in the PRODXA zone.
2. Use the SMPPUNCH output to install the requisite SYSMODs listed in the report.
3. Rerun the REPORT CROSSZONE command for the same ZONESET (SESA) to check for any additional requisites, and install any that are found.
4. Run the REPORT CROSSZONE command for the SXA ZONESET to keep zones PRODXA and MVSXA synchronized.
5. Receive and install SYSMODs as needed for the zones in SXA.
6. Rerun the REPORT CROSSZONE command for SXA, and install additional SYSMODs as needed.

### Example 2: Using REPORT CROSSZONE with Zones Controlled by Different Global Zones

Suppose you want to use the REPORT CROSSZONE command to report on zones controlled by different global zones (in this example, target zones ESA1 and ESA2). To do this, you need to create a new global zone that points to all the zones you want to report on.

1. Allocate a new CSI data set to contain the new global zone. (For information about creating CSI data sets, see the *SMP/E R8.1 User's Guide*.)
2. Define a GLOBALZONE entry in the new CSI data set. This “empty” global zone ties together the zones that are actually controlled from other global zones. (The global zone is “empty” in that it does not contain any actual HOLDDATA or SYSMOD information.)

In this empty global zone, create ZONEINDEX subentries for the zones to be reported on. The zone names must be the same as in the original controlling global zones, and they must be unique in the new global zone. (If the zone names in the controlling global zones match—for example, if the zones you want to compare are both named ESA1—this method will not work.)

Here is an example of the UCLIN statements you can use (make sure your JCL points to the CSI data set containing the new global zone):

```

SET      BDY(GLOBAL)          /* Set to global zone.      */
UCLIN                                /*                          */
ADD      GLOBALZONE          /* Define global now.      */
        SREL(Z038)           /* Identify SRELS.        */
        ZONEINDEX(          /*                          */
                (ESA1, SYS1.ESA1.SMPCSI.CSI, TARGET) /* */
                (ESA2, SYS1.ESA2.SMPCSI.CSI, TARGET) /* */
        )                       /*                          */
        )                       /*                          */
ENDUCL                                /*                          */

```

- In this empty global zone, create a ZONESET entry pointing to the zones to be reported on.

Here is an example of the UCLIN statements you can use to create a ZONESET entry called COMPESA (again, make sure your JCL points to the CSI data set containing the new global zone):

```

SET      BDY(GLOBAL)          /* Set to global zone.      */
UCLIN                                /*                          */
ADD      ZONESET(COMPESA)     /* ZONESET COMPESA.        */
        ZONE(ESA1,          /* Include these target.   */
                ESA2)       /* zones.                  */
        )                       /*                          */
ENDUCL                                /*                          */

```

- Issue the REPORT CROSSZONE command for the ZONESET (and make sure your JCL points to the CSI data set containing the new global zone):

```

SET      BDY(GLOBAL)          /* Process global zone.    */
REPORT  CROSSZONE           /* Report on requisites    */
        ZONESET(COMPESA)     /* for ZONESET COMPESA.   */

```

This example of the REPORT CROSSZONE command does not include the DLIBZONE or TARGETZONE operand, because the ZONESET being used contains only target zones.

This command generates both a report and SMPPUNCH output, which you can use to install the necessary SYSMODs. However, because no SYSMODs have actually been received into the new global zone, the Cross-Zone Requisite SYSMOD report does not indicate which SYSMODs have been received into the original global zones.

## Processing

The REPORT CROSSZONE command checks across zones in a ZONESET for conditional requisites that need to be installed.

SMP/E first verifies that each zone in the ZONESET is defined in the global zone and that all the zones specified on the FORZONE operand are in the ZONESET. Then SMP/E determines which zones will be used.

- If **TZONE** was specified, SMP/E checks that the ZONESET contains target zones and uses those target zones to process the REPORT CROSSZONE command.

- If **DZONE** was specified, SMP/E checks that the ZONESET contains distribution zones and uses those distribution zones to process the REPORT CROSSZONE command.
- If neither **DZONE** nor **TZONE** was specified, SMP/E checks that all the zones in the ZONESET are the same type. If so, it uses all the zones in the ZONESET to process the REPORT CROSSZONE command.

The zones are opened for read access. If **NOPUNCH** was **not** specified, the SMPPUNCH data set is also opened. In addition, if **FORFMID** was specified, the FMIDs and FMIDSETs specified on the FORFMID operand are used to create a list of FMIDs to be reported on.

Next, SMP/E makes a list of all the CIFREQ subentries contained in the zones being used for this REPORT CROSSZONE command. If the FORFMID operand was specified, SMP/E uses only the CIFREQ subentries from SYSMOD entries for functions specified in the FMID list. Otherwise, it lists all the CIFREQ subentries.

**Note:** CIFREQ subentries are only in SYSMOD entries for functions. They list requisites for the function that were specified on an ++IF statement in another SYSMOD. They also list the causer SYSMOD that contained the ++IF statement. For more information about CIFREQ subentries and conditional requisites, see “Conditional Requisites (IFREQ)” on page 32.

SMP/E then checks the CIFREQ list and functions against each of the FORZONE zones (or against all of the zones being used if **FORZONE** was not specified). If the function is installed in the zone (and has not been deleted or superseded) and the causer SYSMOD is not installed there, SMP/E checks whether the requisite is installed in the zone. For each requisite that is not installed, SMP/E writes information for that requisite in the Cross-Zone Requisite SYSMOD report. In addition, commands to install the requisite SYSMODs are written to the SMPPUNCH data set. SMP/E does this processing zone by zone until all the zones to be reported on have been checked. It then notes which of the requisites have not yet been received and writes this information to the SMPRPT data set as well. Finally, it closes all the data sets.

If **any** of the following occurs, SMP/E issues an error message, and REPORT CROSSZONE processing fails:

- The ZONESET or a zone in the ZONESET is not defined in the global zone.
- The zone type in the ZONEINDEX subentry does not match the zone type in the zone definition entry.
- The zones in the ZONESET are not all the same type, and neither **DZONE** nor **TZONE** was specified.
- **TZONE** was specified, but the ZONESET contained no target zones.
- **DZONE** was specified, but the ZONESET contained no distribution zones.
- A zone on the FORZONE operand is not in the ZONESET.
- **TZONE** was specified, and a zone on the FORZONE operand is not a target zone.
- **DZONE** was specified, and a zone on the FORZONE operand is not a distribution zone.

- **NO PUNCH** was not specified, but there is no definition for the SMPPUNCH data set.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of REPORT CROSSZONE processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

- Global zone – Read without enqueue.
- Target zones (as required) – Read without enqueue.
- DLIB zones (as required) – Read without enqueue.

### 2. Processing

- Global zone – Read with shared enqueue.
- Target zones (as required) – Read with shared enqueue.
- DLIB zones (as required) – Read with shared enqueue.

### 3. Termination

- All resources are freed.



---

## Chapter 16. The REPORT ERRSYSMODS Command

This command helps you determine whether any SYSMODs you have already processed are now exception SYSMODs. It also helps you determine whether any resolving SYSMODs are available for held SYSMODs.

- For target and distribution zones, REPORT ERRSYSMODS lists installed SYSMODs for which ++HOLD statements were subsequently received and whose error reason IDs have not yet been resolved.
- For the global zone, REPORT ERRSYSMODS lists received SYSMODs for which ++HOLD statements with error reason IDs have been received.

Information about the held SYSMODs and any resolving SYSMODs is provided in the Exception SYSMOD report. The commands needed to install the resolving SYSMODs are written to the SMPPUNCH data set.

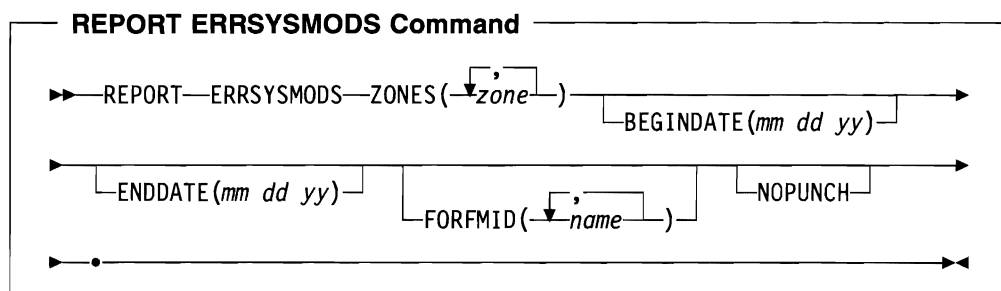
---

### Zones for SET BOUNDARY

For the REPORT ERRSYSMODS command, the SET BOUNDARY command must specify the global zone.

---

### Syntax



---

### Operands

#### BEGINDATE

indicates that you should use ++HOLD statements received by SMP/E Release 5, or later, on or after the specified date for REPORT ERRSYSMODS processing. BEGINDATE can be used alone or with ENDDATE to define the range of the ++HOLD statements to be used.

The date is specified as *mm dd yy*, where *mm* is the month (01–12), *dd* is the day (01–31), and *yy* is the year (00–99). Blanks separate the month, day, and year.

#### Notes:

1. BEGINDATE is allowed only on the REPORT ERRSYSMODS command.
2. If BEGINDATE or ENDDATE is specified, only HOLDDATA that was received using SMP/E Release 5, or later, is processed. To process HOLDDATA that was received using a previous release, do not specify BEGINDATE or ENDDATE.

3. If **BEGINDATE** is specified without **ENDDATE**, SMP/E uses either the **DATE** parameter on the GIMSMP EXEC statement or the current date as the end date.

### **ENDDATE**

indicates that you should use ++HOLD statements received by SMP/E Release 5, or later, on or before the specified date for REPORT ERRSYSMODS processing. **ENDDATE** can be used alone or with **BEGINDATE** to define the range of the ++HOLD statements to be used.

The date is specified as *mm dd yy*, where *mm* is the month (01–12), *dd* is the day (01–31), and *yy* is the year (00–99). Blanks separate the month, day, and year.

#### **Notes:**

1. **ENDDATE** is only allowed on the REPORT ERRSYSMODS command.
2. If **ENDDATE** or **BEGINDATE** is specified, only HOLDDATA that was received using SMP/E Release 5, or later, is processed. To process HOLDDATA that was received using a previous release, do not specify **ENDDATE** or **BEGINDATE**.
3. If **ENDDATE** is specified without **BEGINDATE**, SMP/E processes all HOLDDATA processed by SMP/E Release 5, or later, on or before the specified end date.

### **ERRSYSMODS**

requests a report about SYSMODs for which ++HOLD MCSs with error reason IDs have been received.

- For target and distribution zones, SMP/E reports on SYSMODs that meet **all** these conditions:
  - They have been installed in any of the specified target and distribution zones. (They are not installed in error, have not been deleted, and have not been superseded.)
  - They are in HOLDERERROR status in the global zone. (++HOLD statements with error reason IDs have been received for the SYSMODs.)
  - The error reason IDs are not resolved. (That is, the error reason ID has not been installed, and no SYSMODs that supersede the error reason ID have been installed.)
- For the global zone, SMP/E reports on SYSMODs that meet these conditions:
  - They have been received.
  - They are in HOLDERERROR status in the global zone. (++HOLD statements with error reason IDs have been received for the SYSMODs.)

**ERRSYSMODS** is a required operand for the REPORT ERRSYSMODS command.



**Notes:**

1. ERRSYSMODS is mutually exclusive with CALLLIBS, CROSSZONE, SOURCEID, and SYSMODS.
2. ERRSYSMODS can also be specified as ERRSYS.

**FORFMID**

specifies the FMIDs used to limit which SYSMODs are included in the report. This list can include FMIDs, FMIDSET names, or both.

If **FORFMID** is specified, SMP/E lists only the SYSMODs that have ++HOLD statements with these FMIDs.

If **FORFMID** is not specified, SMP/E reports on all the affected SYSMODs in the specified zones.

**NOPUNCH**

indicates that SMP/E should not write any output to SMPPUNCH. If **NOPUNCH** is **not** specified, JCL or commands are written to SMPPUNCH. This output contains JCL or commands that can be used for further processing.

**Note:** The output produced by REPORT ERRSYSMODS processing contains commands for installing SYSMODs that resolve the error hold reason IDs for exception SYSMODs.

**ZONES**

specifies which zones SMP/E should report on. This list can include zone names, ZONESET names, or both. You can specify the global zone, target zones, distribution zones, or any combination of these. SMP/E produces a separate report for each zone it checks.

For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. For example, suppose you have a ZONESET named MVS1 and a zone named MVS1. If you specify **MVS1** on this operand, SMP/E assumes you want to use the zones defined in ZONESET MVS1 (which might or might not include zone MVS1), and not the individual zone MVS1.

ZONES is a required operand on the REPORT ERRSYSMODS command.

---

**Data Sets Used**

The following data sets may be needed to run the REPORT ERRSYSMODS command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMPCNTL	SMPLOGA	SMPPUNCH	SMPSNAP
SMPCSI	SMPOUT	SMPRPT	<i>zone</i>
SMPLOG			

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

### Usage Notes

If you use the REPORT ERRSYSMODS command, keep these considerations in mind:

- After you run REPORT ERRSYSMODS for target or distribution zones, the Exception SYSMOD report may indicate that some of the resolving SYSMODs are themselves held. To see whether any SYSMODs have been received that resolve these additional holds, run REPORT ERRSYSMODS for the global zone. The resulting Exception SYSMOD report shows any resolving SYSMODs for the additional holds.
- If you run REPORT ERRSYSMODS against the global zone and want to install the additional resolving SYSMODs indicated in the report, you can manually add them to the SMPPUNCH output produced for the affected target or distribution zones.

---

### Output

Output from the REPORT ERRSYSMODS command includes reports, as well as data written to SMPPUNCH.

### Reports

The following reports are produced during REPORT ERRSYSMODS processing:

- Exception SYSMOD report
- File Allocation report

See Chapter 31 for descriptions of these reports.

### SMPPUNCH Output

To make it easier for you to install resolving SYSMODs for exception SYSMODs, SMP/E writes the necessary commands to the SMPPUNCH data set: SET BOUNDARY, RESETRC, and either ACCEPT (for distribution zones) or APPLY (for target zones). Nothing is written to SMPPUNCH for a specified zone in the following cases:

- There are no exception SYSMODs for the specified zone.
- There are no resolving SYSMODs for any of the exception SYSMODs.
- All resolving SYSMODs are held.
- The specified zone is the global zone.
- **NOPUNCH** was specified on the REPORT ERRSYSMODS command.

Figure 12 on page 303 shows the format of the SMPPUNCH output from the REPORT ERRSYSMODS command.

```

SET BDY (zone ).
RESETRC.
command SELECT(
    sysmod1 /* type RESOLVES reason for sysmod2 FMID(sysmod3) */
    *** OR ***
    /* sysmod1 type RESOLVES reason for sysmod2 FMID(sysmod3) */
)
GROUP.

```

Figure 12. REPORT ERRSYSMODS: Format of SMPPUNCH Output

*zone*

is the name of the target or distribution zone in which the exception SYSMOD is currently installed.

*command*

is the command to be used to install the resolving SYSMODs: ACCEPT for a distribution zone, APPLY for a target zone.

*sysmod1*

is the ID of a resolving SYSMOD for the error reason ID.

A SYSMOD is commented out in the following cases:

- The command is ACCEPT and the SYSMOD is an APAR fix. This is to avoid accepting APAR fixes into a distribution zone.
- The SYSMOD is held for an error reason ID.
- The SYSMOD was already listed in the SMPPUNCH output.
- The SYSMOD is a function. This is to avoid inadvertently installing a function.

*type*

is the SYSMOD type of the resolving SYSMOD.

*reason*

is the APAR that caused the exception SYSMOD to be held.

*sysmod2*

is the ID of the exception SYSMOD.

*sysmod3*

is the FMID of the function to which the exception SYSMOD applies.

You can edit the SMPPUNCH output before using it. For example, if a resolving SYSMOD was held and you used REPORT ERRSYSMODS to find a resolving SYSMOD for that hold, you may want to add that second resolving SYSMOD to the SELECT list.

### Example: Using REPORT ERRSYSMODS

Assume you have just received some HOLDDATA, and you need to know whether it affects any of the SYSMODs already accepted into distribution zone DZONE1.

You can use the following commands:

```

SET      BDY(GLOBAL)          /* Process global zone.   */.
REPORT  ERRSYSMODS          /* Report on exception    */.
        ZONES(DZONE1)       /* SYSMODs in this zone  */.
        BEGINDATE(07 01 94) /* for HOLDDATA received */.
        ENDDATE(08 01 94)   /* between these dates.  */.
    
```

Figure 13 is an example of the report SMP/E produces:

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
EXCEPTION SYSMOD REPORT FOR ZONE DZONE1 DATE: 07/01/94 - 08/01/94

NOTE: * - INDICATES THAT THE RESOLVING SYSMOD IS HELD. TO FIND THE RESOLVING
      SYSMODS FOR HELD SYSMODS,
      RUN REPORT ERROR SYSMODS, SPECIFYING THE GLOBAL ZONE

      SYSMOD  HOLD    REASON  RESOLVING ++HOLD ERROR DATA
      NAME    FMID    IDS     SYSMODS

      UZ00001 HBB1200 AZ65432 ***NONE  ++HOLD (UZ00001) FMID(HBB1200)
                                           ERROR REASON(AZ65432)
                                           COMMENT(APAR CAUSES A LOOP).

      JBB1202 AZ00002 AZ00002 ++HOLD (UZ00001) FMID(JBB1202)
      UZ00023* UZ00023* ERROR REASON(AZ00002)
      UZ12090  UZ12090  COMMENT(INCORRECT OUTPUT
                                           DATA SET).

      UZ00325 EJS2409 AZ07003 AZ07003 ++HOLD (UZ00325) FMID(EJS2409)
      EJS2410 EJS2410 ERROR REASON(AZ07003)
                                           COMMENT(SMPPTFIN NOT FOUND).

      UZ64781 JXY0033 AZ07092 AZ07092 ++HOLD (UZ64781) FMID(JXY0033)
      UZ00012 UZ00012 ERROR REASON(AZ07092)
      UZ00023 UZ00023 COMMENT(APAR CAUSES AN
      UZ00204* UZ00204* OC4 ABEND).
      UZ44401 UZ44401
      UZ78003 UZ78003
      AZ60098 ***NONE  ++HOLD (UZ64781) FMID(JXY0033)
                                           ERROR REASON(AZ60098)
                                           COMMENT(INDEFINITE WAIT
                                           STATE).
      AZ60099 UZ00260 ++HOLD (UZ64781) FMID(JXY0033)
      UZ90790 UZ90790 ERROR REASON(AZ60099)
                                           COMMENT(LOG DATA SET NOT
                                           AVAILABLE).
    
```

Figure 13. Example of an Exception SYSMOD Report

SMP/E also writes the commands shown in Figure 14 to the SMPPUNCH data set:

```

SET BDY (DZONE1 ).
RESETRC.
ACCEPT SELECT(
/* AZ00002     APAR     RESOLVES AZ00002 FOR UZ00001 FMID(JBB1202) */
/* UZ00023     PTF      RESOLVES AZ00002 FOR UZ00001 FMID(JBB1202) */
  UZ12090 /* PTF      RESOLVES AZ00002 FOR UZ00001 FMID(JBB1202) */
/* AZ07003     APAR     RESOLVES AZ07003 FOR UZ00325 FMID(EJS2409) */
/* EJS2410     FUNCTION RESOLVES AZ07003 FOR UZ00325 FMID(EJS2409) */
/* AZ07092     APAR     RESOLVES AZ07092 FOR UZ64781 FMID(JXY0033) */
  UZ00012 /* PTF      RESOLVES AZ07092 FOR UZ64781 FMID(JXY0033) */
  UZ00023 /* PTF      RESOLVES AZ07092 FOR UZ64781 FMID(JXY0033) */
/* UZ00204     PTF      RESOLVES AZ07092 FOR UZ64781 FMID(JXY0033) */
  UZ44401 /* PTF      RESOLVES AZ07092 FOR UZ64781 FMID(JXY0033) */
  UZ78003 /* PTF      RESOLVES AZ07092 FOR UZ64781 FMID(JXY0033) */
  UZ00260 /* PTF      RESOLVES AZ07092 FOR UZ64781 FMID(JXY0033) */
  UZ90790 /* PTF      RESOLVES AZ07092 FOR UZ64781 FMID(JXY0033) */
)
GROUP.

```

Figure 14. Example of SMPPUNCH Output for REPORT ERRSYSMODS

After getting the Exception SYSMOD report, you can do the following:

1. Run REPORT ERRSYSMODS, specifying the global zone, to see whether any SYSMODs have been received that resolve the holds for UZ00023 and UZ00204, which are resolving SYSMODs that had holds against them.
2. If the Exception SYSMOD report for the global zone shows resolving SYSMODs for the additional holds, you should edit the SMPPUNCH output to add these new resolving SYSMODs and update the selection list so that UZ00023 and UZ00204 are no longer commented out.
3. Use the SMPPUNCH output to install the resolving SYSMODs in DZONE1.

## Processing

The REPORT ERRSYSMODS command checks the zones specified on the ZONES operand to determine whether SYSMODs that have been processed are now exception SYSMODs.

SMP/E first verifies that each target and distribution zone specified on the ZONES operand is defined in the global zone. The zones and ZONESETs specified on the ZONES operand (including the global zone) are used to create a list of zones to be reported on. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name.

The zones are opened for read access. If **NOPUNCH** was **not** specified, the SMPPUNCH data set is also opened. In addition, if **FORFMID** was specified, the FMIDs and FMIDSETs specified on the FORFMID operand are used to create a list of FMIDs to be reported on.

Next, SMP/E makes a list of all the HOLDDATA entries for error reason IDs. If the FORFMID operand was specified, this HOLDERERROR list contains only HOLDDATA

entries containing ++HOLD statements with an FMID value that is specified in the FMID list. If **BEGINDATE** was specified, the HOLDERERROR list contains entries only for HOLDDATA received by SMP/E Release 5, or later, on or after the specified date. Likewise, if **ENDDATE** was specified, the HOLDERERROR list contains entries only for HOLDDATA received by SMP/E Release 5, or later, on or before the specified date. If neither **BEGINDATE** or **ENDDATE** was specified, the HOLDERERROR list contains all the HOLDDATA entries.

For each zone in the zone list, SMP/E processes the HOLDERERROR list to determine the SYSMODs to be reported on.

- For target and distribution zones, SMP/E reports on SYSMODs that meet these conditions:
  - They have been installed in any of the specified target and distribution zones. (They are not installed in error, have not been deleted, and have not been superseded.)
  - They are in HOLDERERROR status in the global zone. (++)HOLD statements with error reason IDs have been received for the SYSMODs.)
  - The error reason ID is not resolved. (That is, the error reason ID has not been installed, and no SYSMODs that supersede the error reason ID have been installed.)
- For the global zone, SMP/E reports on SYSMODs that meet these conditions:
  - They have been received.
  - They are in HOLDERERROR status in the global zone. (++)HOLD statements with error reason IDs have been received for the SYSMODs.)

After determining which SYSMODs to report on for the specified zone, SMP/E checks the global zone for any resolving SYSMODs that have been received. A SYSMOD resolves an error reason ID if it either matches or specifically supersedes that reason ID. For each exception SYSMOD in the specified zone, SMP/E writes information about that SYSMOD and any resolving SYSMODs in the Exception SYSMOD report. In addition, for target and distribution zones, commands to install the resolving SYSMODs are written to the SMPPUNCH data set. SMP/E does this processing zone by zone until all the zones to be reported on have been checked. Then SMP/E closes all the data sets.

If any of the following occurs, SMP/E issues an error message, and REPORT ERRSYSMODS processing fails:

- A zone, ZONESET, or zone in a ZONESET specified on the ZONES operand (other than the global zone) is not defined in the global zone.
- **NO PUNCH** was not specified, but there is no definition for the SMPPUNCH data set.
- The date range specified by the **BEGINDATE** and **ENDDATE** operands is invalid—for example, if the beginning date is later than the ending date.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of REPORT ERRSYSMODS processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

- Global zone – Read without enqueue.
- Target zones (as required) – Read without enqueue.
- DLIB zones (as required) – Read without enqueue.

### 2. Processing

- Global zone – Read with shared enqueue.
- Target zones (as required) – Read with shared enqueue.
- DLIB zones (as required) – Read with shared enqueue.

### 3. Termination

All resources are freed.





---

## Chapter 17. The REPORT SOURCEID Command

This command helps you determine which source IDs have been assigned to SYSMODs in a given zone or ZONESET.

Information about these source IDs is provided in the SOURCEID report. The commands needed to list the entries for SYSMODs associated with these source IDs are written to the SMPPUNCH data set.

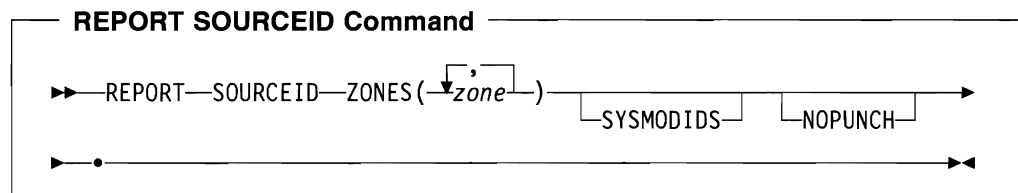
---

### Zones for SET BOUNDARY

For the REPORT SOURCEID command, the SET BOUNDARY command must specify the global zone.

---

### Syntax



---

### Operands

#### **NOPUNCH**

indicates that SMP/E should not write any output to SMPPUNCH. If **NOPUNCH** is **not** specified, JCL or commands are written to SMPPUNCH. This output contains JCL or commands that can be used for further processing.

**Note:** The output produced by REPORT SOURCEID processing contains commands for listing SYSMODs associated with the source IDs that were found in the specified zones.

#### **SOURCEID**

requests a report listing all the source IDs assigned to SYSMODs in a given zone or ZONESET.

SOURCEID is a required operand for the REPORT SOURCEID command.

**Note:** SOURCEID is mutually exclusive with CALLLIBS, CROSSZONE, ERRSYSMODS, and SYSMODS.

#### **SYSMODIDS**

indicates that in the SOURCEID report, SMP/E should also include the IDs of the SYSMODs to which each source ID is assigned.

If **SYSMODIDS** is not specified, the SOURCEID report contains the source IDs assigned to SYSMODs in the zones being reported on, but not the SYSMOD IDs associated with those source IDs.

**Note:** SYSMODIDS is allowed only on the REPORT SOURCEID command.

### ZONES

specifies which zones SMP/E should report on. This list can include zone names, ZONESET names, or both. You can specify the global zone, target zones, distribution zones, or any combination of these. SMP/E produces a separate report for each zone it checks.

For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. For example, suppose you have a ZONESET named MVS1 and a zone named MVS1. If you specify **MVS1** on this operand, SMP/E assumes you want to use the zones defined in ZONESET MVS1 (which might or might not include zone MVS1), and not the individual zone MVS1.

ZONES is a required operand on the REPORT SOURCEID command.

---

### Data Sets Used

The following data sets may be needed to run the REPORT SOURCEID command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMPCNTL	SMPLOGA	SMPPUNCH	SMPSNAP
SMPCSI	SMPOUT	SMRPT	<i>zone</i>
SMLOG			

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

---

### Output

Output from the REPORT SOURCEID command includes reports, as well as data written to SMPPUNCH.

### Reports

The following reports are produced during REPORT SOURCEID processing:

- SOURCEID report
- File Allocation report

See Chapter 31 for descriptions of these reports.

### SMPPUNCH Output

To make it easier for you to list the SYSMODs associated with the SOURCEIDs named in the SOURCEID report, SMP/E writes the necessary commands to the SMPPUNCH data set: SET BOUNDARY, RESETRC, and LIST SYSMOD SOURCEID(...). Nothing is written to SMPPUNCH for a specified zone in the following cases:

- There are no SYSMOD entries in the zone.
- There are no source IDs assigned to SYSMODs in the zone.
- **NOPUNCH** was specified on the REPORT SOURCEID command.

Figure 15 shows the format of the SMPPUNCH output from the REPORT SOURCEID command.

```

SET BDY(zonename).
RESETRC.
LIST SYSMOD SOURCEID(
                    sourceid
                    sourceid
                    ).

SET BDY(zonename).
RESETRC.
LIST SYSMOD SOURCEID(
                    sourceid
                    sourceid
                    ).

```

Figure 15. REPORT SOURCEID: Format of SMPPUNCH Output

*zonename*

is the name of a zone specified on the ZONES operand as an individual zone or a member of a ZONESET.

*sourceid*

is the source ID assigned to a SYSMOD in the specified zone.

You can edit the SMPPUNCH output before using it.

## Examples

The following examples are provided to help you use the REPORT SOURCEID command:

### Example 1: REPORT SOURCEID (SYSMODIDS Operand Specified)

Assume you want to find out which source IDs are associated with SYSMODs in target zone TGT1, and you want to know which SYSMODs each source ID is assigned to. You can use the following commands:

```

SET BDY(GLOBAL).
REPORT SOURCEID
      ZONES(TGT1)
      SYSMODIDS.

```

Figure 16 is an example of the report SMP/E writes:

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

SOURCEID REPORT FOR TARGET ZONE TGT1

NOTE: * - INDICATES THE SYSMOD HAS MULTIPLE SOURCEIDS

SOURCEID      _____ SYSMOD NAMES _____
PUT9303      UZ00023*  UZ00024  UZ00035  UZ00037  UZ00039
              UZ00052  UZ00073  UZ00076  UZ00077
SMCREC       UZ00015  UZ00023*  UZ00044
    
```

Figure 16. Example of a SOURCEID Report (SYSMODIDS Operand Specified)

SMP/E also writes the commands shown in Figure 17 to the SMPPUNCH data set:

```

SET BDY(TGT1).
RESETRC.
LIST SYSMOD SOURCEID(
                        PUT9303
                        SMCREC
                        ).
    
```

Figure 17. Example of SMPPUNCH Output for REPORT SOURCEID (SYSMODIDS Operand Specified)

**Example 2: REPORT SOURCEID (SYSMODIDS Operand Not Specified)**

Assume you want to find out the source IDs associated with SYSMODs in target zone TGT2, but you are not interested in knowing the specific SYSMODs that each SOURCEID is assigned to. You can use the following commands:

```

SET BDY(GLOBAL).
REPORT SOURCEID
      ZONES(TGT2).
    
```

Figure 18 is an example of the report SMP/E writes:

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

SOURCEID REPORT FOR TARGET ZONE TGT2

_____ SOURCEIDS _____

PUT9206  PUT9207  PUT9208  PUT9301  PUT9302  PUT9303
PUT9304
    
```

Figure 18. Example of a SOURCEID Report (SYSMODIDS Operand Not Specified)

SMP/E also writes the commands shown in Figure 19 to the SMPPUNCH data set:

```

SET BDY(TGT2).
RESETRC.
LIST SYSMOD SOURCEID(
                                PUT9206
                                PUT9207
                                PUT9208
                                PUT9301
                                PUT9302
                                PUT9303
                                PUT9304
                                ).

```

Figure 19. Example of SMPPUNCH Output for REPORT SOURCEID (SYSMODIDS Operand Not Specified)

**Example 3: REPORT SOURCEID (ZONES Operand Specified)**

Assume you want to find out the source IDs associated with SYSMODs in DLB3, plus all the zones defined by ZONESET MVSPROD3. (MVSPROD3 defines two zones: DLB3 and TGT3.) You are not interested in knowing the specific SYSMODs each source ID is assigned to. You can use the following commands:

```

SET BDY(GLOBAL).
REPORT SOURCEID
      ZONES(DLB3, MVSPROD3).

```

SMP/E writes one report for each zone specified by the ZONES operand. Even though DLB3 is specified individually and also included by ZONESET MVSPROD3, it is reported on only once.

Figure 20 shows examples of the reports SMP/E would write for DLB3 and TGT3:

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
SOURCEID REPORT FOR DLIB ZONE DLB3
_____SOURCEIDS_____
PUT9206 PUT9207 PUT9208 PUT9301 PUT9302 PUT9303
PUT9304

```

Figure 20 (Part 1 of 2). Example of SOURCEID Reports (ZONES Operand Specified)

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
SOURCEID REPORT FOR TARGET ZONE TGT3
_____SOURCEIDS_____
PUT9206 PUT9207 PUT9208 PUT9301 PUT9302
```

Figure 20 (Part 2 of 2). Example of SOURCEID Reports (ZONES Operand Specified)

SMP/E also writes the commands shown in Figure 21 to the SMPPUNCH data set:

```
SET BDY(DLB3).
RESETRC.
LIST SYSMOD SOURCEID(
                PUT9206
                PUT9207
                PUT9208
                PUT9301
                PUT9302
                PUT9303
                PUT9304
                ).
SET BDY(TGT3).
RESETRC.
LIST SYSMOD SOURCEID(
                PUT9206
                PUT9207
                PUT9208
                PUT9301
                PUT9302
                ).
```

Figure 21. Example of SMPPUNCH Output for REPORT SOURCEID (ZONES Operand Specified)

---

## Processing

The REPORT SOURCEID command checks the SYSMOD entries in a zone and reports on any source IDs found in those entries. The resulting output can be used to list the SYSMOD entries associated with the source IDs that were found.

SMP/E first verifies that each target and distribution zone specified on the ZONES operand is defined in the global zone. The zones and ZONESETs specified on the ZONES operand (including the global zone) are used to create a list of zones to be reported on. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a zone is specified separately and is also part of a ZONESET, it is only reported on once.

The zones are opened for read access. If **NOPUNCH** was **not** specified, the SMPPUNCH data set is also opened.

For each zone to be reported on, SMP/E checks all the SYSMOD entries in that zone to see if they contain source IDs.

- For each SYSMOD entry containing a source ID, SMP/E saves the SOURCEID value and the SYSMOD ID.
- If a SYSMOD entry contains multiple source IDs, SMP/E saves all its SOURCEID values. It also saves an indicator so the SOURCEID report can note that the SYSMOD has multiple source IDs.
- If a SYSMOD is in error, it is not considered installed. As a result, SMP/E ignores any source IDs associated with that SYSMOD.

SMP/E then writes a SOURCEID report for each zone specified by the ZONES operand.

- If **SYSMODIDS** was specified on the REPORT SOURCEID command, the report includes the IDs of the SYSMODs associated with each SOURCEID value found in the zone.
- If **SYSMODIDS** was not specified on the REPORT SOURCEID command, the report includes all the SOURCEID values found in the zone, but not the IDs of the SYSMODs associated with those source IDs.

In addition, if **NOPUNCH** was **not** specified, SMP/E writes commands to the SMPPUNCH data set to list the SYSMODs associated with the source IDs that were found. Finally, SMP/E closes all the data sets.

If any of the following occurs, SMP/E issues an error message, and REPORT SOURCEID processing fails:

- A zone, ZONESET, or zone in a ZONESET specified on the ZONES operand (other than the global zone) is not defined in the global zone.
- **NOPUNCH** was not specified, but there is no definition for the SMPPUNCH data set.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of REPORT SOURCEID processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

- |                            |   |                       |
|----------------------------|---|-----------------------|
| Global zone                | – | Read without enqueue. |
| Target zones (as required) | – | Read without enqueue. |
| DLIB zones (as required)   | – | Read without enqueue. |

### 2. Processing

- |                            |   |                           |
|----------------------------|---|---------------------------|
| Global zone                | – | Read with shared enqueue. |
| Target zones (as required) | – | Read with shared enqueue. |
| DLIB zones (as required)   | – | Read with shared enqueue. |

3. Termination

All resources are freed.





---

## Chapter 18. The REPORT SYSMODS Command

This command helps you compare the SYSMODs installed in two zones. These are the types of zones you can compare:

- A DLIB zone to a DLIB zone
- A target zone to a target zone
- A DLIB zone to a target zone
- A target zone to a DLIB zone

Information about the SYSMODs is provided in the SYSMOD Comparison report. The commands needed to install SYSMODs in one zone, but not in the other, are written to the SMPPUNCH data set.

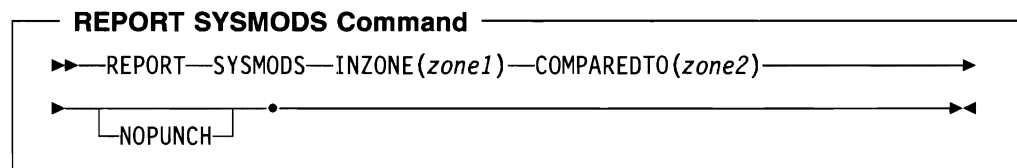
---

### Zones for SET BOUNDARY

For the REPORT SYSMODS command, the SET BOUNDARY command must specify the global zone.

---

### Syntax



### Operands

#### COMPAREDTO

specifies the zone (called the *comparison zone*) that SMP/E should compare against the input zone for SYSMOD content. You can specify a single target zone or distribution zone name. This must **not** be the same as the zone specified on the INZONE operand. A report is issued to indicate which SYSMODs were in the input zone but not in the comparison zone.

COMPAREDTO is a required operand on the REPORT SYSMODS command.

**Note:** COMPAREDTO is allowed only on the REPORT SYSMODS command.

#### INZONE

specifies the input zone that SMP/E should use to compare against another zone (called the *comparison zone*) for SYSMOD content. You can specify a single target zone or distribution zone name. This must **not** be the same as the zone specified on the COMPAREDTO operand. A report is issued to indicate which SYSMODs were in the input zone but not in the comparison zone.

INZONE is a required operand on the REPORT SYSMODS command.

**Note:** INZONE is allowed only on the REPORT SYSMODS command.

### NOPUNCH

indicates that SMP/E should not write any output to SMPPUNCH. If **NOPUNCH** is **not** specified, JCL or commands are written to SMPPUNCH. This output contains JCL or commands that can be used for further processing.

**Note:** The output produced by REPORT SYSMODS processing contains commands for installing SYSMODs from the input zone that are applicable to the comparison zone.

### SYSMODS

requests a report comparing the SYSMOD content of two zones.

SYSMODS is a required operand for the REPORT SYSMODS command.

**Note:** SYSMODS is mutually exclusive with CALLLIBS, CROSSZONE, SOURCEID, and ERRSYSMODS.

---

## Data Sets Used

The following data sets may be needed to run the REPORT SYSMODS command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMPCNTL	SMPLOGA	SMPPUNCH	SMPSNAP
SMPCSI	SMPOUT	SMPRPT	<i>zone</i>
SMPLOG			

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

---

## Output

Output from the REPORT sysmods command includes reports, as well as data written to SMPPUNCH.

## Reports

The following reports are produced during REPORT SYSMODS processing:

- SYSMOD Comparison report
- File Allocation report

See Chapter 31 for descriptions of these reports.

## SMPPUNCH Output

To make it easier for you to install the applicable SYSMODs named in the SYSMOD Comparison report, SMP/E writes the necessary commands to the SMPPUNCH data set: SET BOUNDARY, RESETRC, and either ACCEPT (for distribution zones) or APPLY (for target zones). Nothing is written to SMPPUNCH for a specified zone in the following cases:

- There are no applicable SYSMODs in the input zone.
- **NOPUNCH** was specified on the REPORT SYSMODS command.

Figure 22 on page 319 shows the format of the SMPPUNCH output from the REPORT SYSMODS command.

```
SET BDY (zone2 ).  
RESETRC      /*
```

THE FOLLOWING *command* COMMAND(S) WERE GENERATED BY A REPORT SYSMODS COMMAND ON *yy.ddd* AT *hh:mm:ss*. THE SELECTED SYSMODS WERE FOUND IN THE INPUT ZONE (*zone1\_*) BUT NOT IN THE COMPARISON ZONE (*zone2\_*) AND APPEAR APPLICABLE TO THE COMPARISON ZONE.

SYSMODS IN THE SELECT LIST ARE GROUPED BY FMID. FUNCTIONS APPEAR FIRST FOLLOWED BY SERVICE SYSMODS (PTFS, APARS, AND USERMODS IN THAT ORDER).

FUNCTIONS AND PTFS THAT WERE AVAILABLE IN THE GLOBAL ZONE AND ARE APPLICABLE TO THE COMPARISON ZONE APPEAR AS NORMAL SELECT LIST VALUES WITH AN APPROPRIATE COMMENT. FUNCTIONS AND PTFS THAT WERE NOT AVAILABLE IN THE GLOBAL ZONE OR HAVE UNKNOWN APPLICABILITY APPEAR AS SMP COMMENTS. PTFS FOR AN APPLICABLE FUNCTION, NOT AVAILABLE IN THE GLOBAL ZONE, APPEAR AS COMMENTS EVEN IF THEY ARE AVAILABLE. SUCH COMMENTED SYSMODS CAN BE CHANGED TO NORMAL SELECT LIST VALUES WHEN THEY OR THEIR APPLICABLE FUNCTIONS ARE RECEIVED OR ARE DETERMINED TO BE APPLICABLE.

APARS AND USERMODS APPEAR AS SMP COMMENTS. THESE COMMENTED SYSMODS CAN BECOME NORMAL SELECT LIST VALUES IF YOU DESIRE TO INSTALL THEM IN THE *zone2\_* ZONE AND THEY ARE AVAILABLE IN THE GLOBAL ZONE. THE COMMENTARY TEXT INDICATES THEIR AVAILABILITY AND APPLICABILITY STATUS.

THE COMMAND(S) WERE GENERATED WITH THE GROUP AND CHECK OPTIONS. GROUP CAUSES ANY REQUISITE SYSMODS TO BE INCLUDED ALONG WITH THE SELECTED SYSMODS. CHECK CAUSES A DRYRUN OF THE COMMAND BEFORE LIBRARIES ARE UPDATED SO THAT ANY NEEDED CORRECTIVE ACTION CAN BE TAKEN BEFORE REAL INSTALLATION.

REDO WAS GENERATED IF A FUNCTION IS TO BE REINSTALLED. IF REDO IS USED, ONLY THE FUNCTION BEING REINSTALLED AND ITS SERVICE ARE SELECTED ON THAT COMMAND.

\*\*\*\* WARNING \*\*\*\*

IT IS POSSIBLE THAT ALL SYSMODS SELECTED WILL APPEAR AS SMP COMMENTS. IF THIS IS THE CASE, SMP WILL ISSUE A SYNTAX ERROR IF YOU RUN THE GENERATED COMMAND(S) AS IS.

\*\*\*\* WARNING \*\*\*\*

\*/

Figure 22 (Part 1 of 2). REPORT SYSMODS: Format of SMPPUNCH Output

```

command
SELECT(
  sysm did      /* smdtype FOR fmid RECEIVED                */
/* sysm did      smdtype FOR fmid NOT RECEIVED            */
/* sysm did      smdtype FOR fmid RECEIVED, fmid NOT RECV'D  */
/* sysm did      smdtype FOR fmid NOT RECEIVED, fmid NOT RECV'D */
/* sysm did      smdtype FOR fmid RECEIVED, APPLICABILITY UNKNOWN */
/* sysm did      smdtype FOR fmid NOT RECV'D, APPLICABILITY UNKNOWN */
/* sysm did      APAR    FOR fmid RECEIVED                */
/* sysm did      APAR    FOR fmid NOT RECEIVED            */
/* sysm did      USERMOD FOR fmid RECEIVED                */
/* sysm did      USERMOD FOR fmid NOT RECEIVED            */
)

[REDO]
GROUP
CHECK.

```

Figure 22 (Part 2 of 2). REPORT SYSMODS: Format of SMPPUNCH Output

*zone2*

is the name of the target or distribution zone specified on the COMPAREDTO operand (the comparison zone).

*command*

is the command to be used to install the SYSMODs: ACCEPT for a distribution zone, and APPLY for a target zone.

*yy.ddd*

is the year and Julian date that the command was generated by the REPORT SYSMODS command.

*hh:mm:ss*

is the time of day at which the command was generated by the REPORT SYSMODS command.

*zone1*

is the name of the target or distribution zone specified on the INZONE operand (the input zone).

*sysm did*

is the ID of an applicable SYSMOD that appears in the SYSMOD Comparison report.

A SYSMOD is commented out in the following cases:

- The SYSMOD is an APAR fix or a USERMOD.
- The SYSMOD or its owning function (FMID) is not in the global zone.
- SMP/E cannot determine whether the SYSMOD is applicable to the comparison zone.

*smdtype*

is the SYSMOD type of the indicated SYSMOD.

*fmid*

is the SYSMOD ID of the function that owns the indicated SYSMOD.

You can edit the SMPPUNCH output before using it. For example, if any applicable SYSMODs were not in the global zone at the time of the report and you have since received them, you may want to change the comments for those SYSMODs to normal SELECT list values and install them.

**Note:** There may have been SYSMODs whose applicability was unknown (for example, they were not in the global zone), and which you have determined to be applicable. You may want to receive these SYSMODs and change them from comments to normal select list values so they can be installed.

You can determine whether a SYSMOD is applicable to the comparison zone by checking the program directory or installation manual for the FMID to find out the SREL, or you can receive the SYSMOD into the global zone and run the REPORT SYSMODS command again.

Multiple APPLY or ACCEPT commands may be generated if one or more functions are being reinstalled. Each function SYSMOD to be reinstalled appears on a separate APPLY or ACCEPT command along with its own service. Other SYSMODs belonging to functions not being reinstalled appear on a separate APPLY or ACCEPT command.

---

## Example: Using REPORT SYSMODS

Assume you have two MVS/ESA systems. The target zones controlling these systems are MVSESA1 and MVSESA2, and they are serviced from the same global zone. You want to determine which SYSMODs are installed in MVSESA1 and are not installed in, but are applicable to, MVSESA2. The following chart shows the SYSMODs installed in the MVSESA1 and MVSESA2 zones and the SYSMODs available in the global zone at the time the REPORT SYSMODS command is run.

**Note:** All the zones have a single SREL value of Z038. The SYSMODs for all the zones are annotated with information about their FMID, source IDs, and installation status. These are the meanings of the annotations:

- **DELBY** indicates that the SYSMOD has not been installed, but its SYSMOD entry contains the DELBY subentry. The entry was created when a function was installed that deleted the indicated SYSMOD.
- **ERROR** indicates that the SYSMOD has only been partially installed. Errors occurred during APPLY or ACCEPT processing.
- **SUPONLY** indicates that the SYSMOD is a superseded-only SYSMOD.
- **F=** is followed by the SYSMOD's FMID.
- **S=** is followed by the SYSMOD's source IDs.

*Table 22. REPORT SYSMODS Example: SYSMODs Installed in Each Zone*

Zone	Functions	PTFs	APARs	USERMODs
MVSESA1	HAA1202 HBB1102 DELBY HBB1202 JBB1122 SUPONLY JBB1222 F=HBB1202	UZ00001 F=HBB1202 S=PUT9206 UZ00002 F=HBB1202 S=PUT9207 PDO0001 UZ00011 F=HAA1202 S=PUT9206 UZ00012 F=HAA1202 S=PUT9207 UZ00013 F=HAA1202 S=PUT9207 ERROR UZ00021 F=JBB1222 S=PUT9206 UZ00022 F=JBB1222 S=PUT9207	AZ00001 SUPONLY AZ00002 SUPONLY AZ00011 SUPONLY AZ00012 SUPONLY AZ00013 F=HAA1202 S=RETAIN AZ00021 SUPONLY AZ00022 SUPONLY	TZ00001 F=HAA1202
MVSESA2	HBB1202 JBB1122 SUPONLY JBB1222 SUPONLY JBB1322 F=HBB1202	UZ00001 F=HBB1202 S=PUT9206 UZ00031 F=JBB1322 S=PUT9206 UZ00032 F=JBB1322 S=PUT9207	AZ00001 SUPONLY AZ00031 SUPONLY AZ00032 SUPONLY	None
GLOBAL	HAA1202 JBB1322 F=HBB1202	UZ00002 F=HBB1202 S=PUT9207 UZ00012 F=HAA1202 S=PUT9207 UZ00022 F=JBB1222 S=PUT9207 UZ00031 F=JBB1322 S=PUT9206 UZ00032 F=JBB1322 S=PUT9207	None	TZ00001 F=HAA1202

Assume you want to find out which SYSMODs are installed in zone MVSESA1 and are not installed in zone MVSESA2, but are applicable to it. You can use the following commands:

```
SET    BDY(GLOBAL)          /* Process global zone.  */.
REPORT SYSMODS              /* Report on SYSMODs.    */.
      INZONE(MVSESA1)      /* Input zone MVSESA1.   */.
      COMPAREDTO(MVSESA2) /* Comparison zone MVSESA2.*/.
```

SMP/E compares the SYSMOD content of zone MVSESA1 to that of zone MVSESA2. Any SYSMODs that are in zone MVSESA1 (with the exceptions noted under "Processing" on page 325) and are not in zone MVSESA2 appear in the resulting report.

Figure 23 shows the report SMP/E produces:

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
SYSMOD COMPARISON REPORT FOR TARGET ZONE MVSESA1 AND TARGET ZONE MVSESA2
MATCHING SREL(S) - Z038
```

FMID__	SYSMOD_	TYPE__	APPLICABLE	RECEIVED	SOURCEIDS
HAA1202	HAA1202	FUNCTION	YES	YES	
	UZ00011	PTF	YES	NO	PUT9206
	UZ00012	PTF	YES	YES	PUT9207
	AZ00013	APAR	YES	NO	RETAIN
	TZ00001	USERMOD	YES	YES	
HBB1202	UZ00002	PTF	YES	YES	PUT9207
					PD00001
JBB1222	UZ00021	PTF	NO	NO	PUT9206
	UZ00022	PTF	NO	YES	PUT9207

Figure 23. Example of a SYSMOD Comparison Report

SMP/E also writes the commands shown in Figure 24 to the SMPPUNCH data set:

```
SET BDY (MVSESA2).  
RESETRC      /*
```

THE FOLLOWING APPLY COMMAND(S) WERE GENERATED BY A REPORT SYSMODS COMMAND ON *yy.ddd* AT *hh:mm:ss*. THE SELECTED SYSMODS WERE FOUND IN THE INPUT ZONE (MVSESA1) BUT NOT IN THE COMPARISON ZONE (MVSESA2) AND APPEAR APPLICABLE TO THE COMPARISON ZONE.

SYSMODS IN THE SELECT LIST ARE GROUPED BY FMID. FUNCTIONS APPEAR FIRST FOLLOWED BY SERVICE SYSMODS (PTFS, APARS, AND USERMODS IN THAT ORDER).

FUNCTIONS AND PTFS THAT WERE AVAILABLE IN THE GLOBAL ZONE AND ARE APPLICABLE TO THE COMPARISON ZONE APPEAR AS NORMAL SELECT LIST VALUES WITH AN APPROPRIATE COMMENT. FUNCTIONS AND PTFS THAT WERE NOT AVAILABLE IN THE GLOBAL ZONE OR HAVE UNKNOWN APPLICABILITY APPEAR AS SMP COMMENTS. PTFS FOR AN APPLICABLE FUNCTION, NOT AVAILABLE IN THE GLOBAL ZONE, APPEAR AS COMMENTS EVEN IF THEY ARE AVAILABLE. SUCH COMMENTED SYSMODS CAN BE CHANGED TO NORMAL SELECT LIST VALUES WHEN THEY OR THEIR APPLICABLE FUNCTIONS ARE RECEIVED, OR ARE DETERMINED TO BE APPLICABLE.

APARS AND USERMODS APPEAR AS SMP COMMENTS. THESE COMMENTED SYSMODS CAN BECOME NORMAL SELECT LIST VALUES IF YOU DESIRE TO INSTALL THEM IN THE MVSESA2 ZONE AND THEY ARE AVAILABLE IN THE GLOBAL ZONE. THE COMMENTARY TEXT INDICATES THEIR AVAILABILITY AND APPLICABILITY STATUS.

THE COMMAND(S) WERE GENERATED WITH THE GROUP AND CHECK OPTIONS. GROUP CAUSES ANY REQUISITE SYSMODS TO BE INCLUDED ALONG WITH THE SELECTED SYSMODS. CHECK CAUSES A DRYRUN OF THE COMMAND BEFORE LIBRARIES ARE UPDATED SO THAT ANY NEEDED CORRECTIVE ACTION CAN BE TAKEN BEFORE REAL INSTALLATION.

REDO WAS GENERATED IF A FUNCTION IS TO BE REINSTALLED. IF REDO IS USED, ONLY THE FUNCTION BEING REINSTALLED AND ITS SERVICE ARE SELECTED ON THAT COMMAND.

Figure 24 (Part 1 of 2). Example of SMPPUNCH Output for REPORT SYSMODS



```

          **** WARNING ****

IT IS POSSIBLE THAT ALL SYSMODS SELECTED WILL APPEAR AS SMP
COMMENTS. IF THIS IS THE CASE, SMP WILL ISSUE A SYNTAX ERROR IF YOU
RUN THE GENERATED COMMAND(S) AS IS.

          **** WARNING ****

          .
          */

APPLY
SELECT(
  HAA1202    /* FUNCTION FOR HAA1202 RECEIVED          */
/* UZ00011   PTF      FOR HAA1202 NOT RECEIVED        */
  UZ00012    /* PTF      FOR HAA1202 RECEIVED          */
/* AZ00013   APAR     FOR HAA1202 NOT RECEIVED        */
/* TZ00001   USERMOD  FOR HAA1202 RECEIVED          */
  UZ00002    /* PTF      FOR HBB1202 RECEIVED          */
)

GROUP
CHECK.

```

Figure 24 (Part 2 of 2). Example of SMPPUNCH Output for REPORT SYSMODS

After getting the SYSMOD Comparison report, you can do the following:

1. Research the report to determine which of the identified SYSMODs you want to install into the comparison zone.
2. Find and receive any applicable SYSMODs that were not available and that you want to install. The source ID in the report identifies some possible sources for obtaining the SYSMODs.
3. Tailor the SMPPUNCH output to install the set of SYSMODs that you deem appropriate, and run the commands to install the desired SYSMODs.

## Processing

The REPORT SYSMODS command compares the SYSMOD content of an input target or distribution zone to that of a comparison target or distribution zone. The resulting output can be used to determine which SYSMODs might need to be installed in the comparison zone to make its function and service content more equivalent to that of the input zone.

SMP/E first checks the REPORT SYSMODS command to determine the zones to be compared and whether SMPPUNCH output should be produced. The zones are then opened for read access along with the global zone. If **NOPUNCH** was **not** specified, the SMPPUNCH data set is also opened.

Next, SMP/E determines the matching SREL values from the zone definitions of the zones being compared. Matching values are saved to be put in the header of the SYSMOD Comparison report.

SMP/E then determines whether or not the zones to be compared contain SYSMOD entries. If they both contain SYSMOD entries, SMP/E looks for SYSMODs that are installed in the input zone but not in the comparison zone, and checks whether those SYSMODs are applicable to the comparison zone.

1. First, SMP/E reads sequentially through the SYSMOD entries in the input zone to determine which SYSMODs should be included in the report. For each SYSMOD entry that is **not** a superseded-only entry or deleted-only entry and is not in error, SMP/E checks the comparison zone to see if the SYSMOD also exists there. The SYSMOD exists in the comparison zone if **one** of the following is true:
  - There is a SYSMOD entry with the ERROR indicator off.
  - There is a SYSMOD entry with the DELBY subentry.
  - There is a SYSMOD entry with the SUPBY subentry.

If the SYSMOD does **not** exist in the comparison zone, SMP/E includes it in the SYSMOD Comparison report.

If the SYSMOD **exists** in the comparison zone and is a function, it may still be included in the SYSMOD Comparison report. This can happen when the function in the input zone has been service-updated. A service-updated function may be at a higher service level than the function currently installed in the comparison zone. Therefore, SMP/E does additional checking for a function that exists in both the input and comparison zones.

If a function in the input zone has been service-updated, it supersedes the service integrated into it. To determine whether a service-updated function should be reinstalled in the comparison zone, SMP/E checks whether all the SYSMODs superseded by the function exist in the comparison zone. If any of the superseded SYSMODs do **not** exist in the comparison zone, the service-updated function can be reinstalled and is included in the report.

**Note:** When SMP/E checks the global zone to determine whether a service-update is available to be reinstalled, it also checks the superseded information in the global zone SYSMOD entry against the input zone SYSMOD entry. This is done to ensure that the function in the global zone is not at a lower service level than the SYSMOD in the input zone.

2. Next, SMP/E determines whether the SYSMODs included in the report are applicable to the comparison zone.
  - If the SYSMOD is a service-updated function that can be reinstalled, it is applicable to the comparison zone.
  - If the SYSMOD is a base function, it is applicable to the comparison zone if it meets **either** of these conditions:
    - All the SRELs supported by the input zone are also supported by the comparison zone.
    - Any of the SRELs defined for the SYSMOD are supported by the comparison zone.

**Note:** SMP/E checks the global zone SYSMOD entry to determine which SRELs are defined for the SYSMOD. If there is no entry for the SYSMOD in the global zone, SMP/E cannot determine the SYSMOD's SREL.

If there is at least one SREL in the input zone that is **not** in the comparison zone and there is no SYSMOD entry in the global zone, the SYSMOD **may** or **may not** be applicable to the comparison zone. In this case, SMP/E indicates in the report that the applicability of the SYSMOD is unknown.

- If the SYSMOD is not a base function, it is applicable to the comparison zone if its FMID meets **either** of these conditions:
  - It exists in the comparison zone.
  - It has already been deemed applicable during this run of the REPORT SYSMODS command. (If the applicability of the FMID is unknown, the applicability of this SYSMOD is also unknown.)

At this point, SMP/E gathers additional information about the SYSMOD to include in the SYSMOD Comparison report:

- It takes the FMID, SYSMOD type (function, PTF, APAR, or USERMOD), and source IDs from the SYSMOD entry in the input zone.
- It checks whether there is an entry for the SYSMOD in the global zone.

SMP/E saves all the information gathered in the above processing and uses it in the SYSMOD Comparison report and SMPPUNCH output after checking all the SYSMOD entries in the input zone. Finally, it closes all the data sets.

SMP/E issues an error message, and REPORT SYSMODS processing fails if any of the following occurs:

- The zone specified on the INZONE or COMPAREDTO operand is not defined in the global zone.
- The zones specified on the INZONE and COMPAREDTO operands have no matching SREL values in their zone definitions.
- The zone specified on the INZONE or COMPAREDTO operand is the global zone.
- The zone specified on the INZONE or COMPAREDTO operand contains no SYSMOD entries.
- The zones specified on the INZONE and COMPAREDTO operands are the same.
- **NOPUNCH** was not specified, but there is no definition for the SMPPUNCH data set.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of REPORT SYSMODS processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

1. Initialization

- Global zone – Read without enqueue.
- Target zones (as required) – Read without enqueue.
- DLIB zones (as required) – Read without enqueue.

2. Processing

- Global zone – Read with shared enqueue.
- Target zones (as required) – Read with shared enqueue.
- DLIB zones (as required) – Read with shared enqueue.

3. Termination

All resources are freed.

---

## Chapter 19. The RESETRC Command

Many SMP/E commands depend on the successful processing of preceding commands. To help avoid errors resulting from the failure of preceding commands, SMP/E saves the highest return code issued for each command. As it processes each command, it checks these return codes to make sure all the preceding commands succeeded.

At times, you may want to process commands that do not depend on the success of any preceding commands. To do this, you can use the RESETRC command. RESETRC sets the return codes for the preceding commands to zero, thus allowing SMP/E to process the current command.

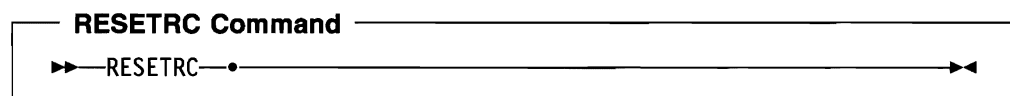
---

### Zones for SET BOUNDARY

The RESETRC command is used only to determine whether subsequent commands are to be processed. Therefore, you should use the same SET BOUNDARY command for RESETRC as for the subsequent commands.

---

### Syntax



---

### Data Sets Used

The following data sets may be needed to run the RESETRC command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOG	SMPOUT	SMPSNAP
SMP_CSI	SMPLOGA		

---

### Usage Notes

- You should **not** use RESETRC before commands that depend on the success of preceding commands.
- RESETRC only resets return codes issued for SMP/E commands. It does not affect the maximum return code set when SMP/E itself fails. That return code is always set to the highest return code issued for any command processed during that calling of SMP/E.
- There is no return code for the RESETRC command.

## Examples

The following examples are provided to help you use the RESETRC command:

- “Example 1: Using RESETRC between Commands for One Zone”
- “Example 2: Using RESETRC between Commands for Different Zones”

### Example 1: Using RESETRC between Commands for One Zone

The processing of one command may **not** depend on the success of preceding commands if you divide the work that is to be done to a single zone into multiple steps. For example, you might use the source ID and FMIDSET operands to apply SYSMODs for two different functions. In this example, you want to install two groups of PTFs from service level PUT9301. One group is for function EBB1102, and the other is for an unrelated function, EDM1102. To prevent possible errors from one APPLY command from affecting the processing of the other, you can put a RESETRC command between the two APPLY commands:

```

SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone.*/.
APPLY    SOURCEID(PUT9301) /* Apply service level 9301 */
          FORFMID(EBB1102) /* for function EBB1102.   */.
RESETRC                                     /* Next set of commands
                                     not dependent on
                                     successful apply of
                                     EBB1102 service.      */.
APPLY    SOURCEID(PUT9301) /* Now apply service for   */
          FORFMID(EDM1102) /* EDM1102.                */.
    
```

SMP/E first applies PTFs from service level 9301 that are for function EBB1102. It then applies PTFs from service level 9301 that are for function EDM1102. Without the intervening RESETRC command, the second APPLY runs only if the return code for the first APPLY was less than 12.

### Example 2: Using RESETRC between Commands for Different Zones

You might also want to use the RESETRC command when you are installing the same changes on different systems. If the two systems are not identical, errors that might occur during installation on one system may not occur during installation on the other system. For example, assume you want to install two PTFs that are applicable to two different systems, MVSTST1 and MVSTST2. To prevent possible errors from the SET or APPLY commands for one system from affecting processing in the other system, you can put a RESETRC command between the two groups of SET and APPLY commands:

```

SET      BDY(MVSTST1)      /* Set to process MVSTST1. */.
APPLY    S(UR12345,UR12346) /* Apply two PTFs.        */.
RESETRC                                     /* Next set of commands is
                                     not dependent on successful
                                     apply of two PTFs.      */.
SET      BDY(MVSTST2)      /* Set to process MVSTST2. */.
APPLY    S(UR12345,UR12346) /* Apply two PTFs.        */.
    
```

SMP/E first applies the two PTFs to MVSTST1, and then applies the same PTFs to MVSTST2. Without the intervening RESETRC command, the second group of SET and APPLY commands runs only if the return code for the first APPLY was less than 12.



## Processing

For each command processed in a single invocation of SMP/E, SMP/E keeps a record of the highest return code for that command. As SMP/E processes each command, it checks the saved return codes from all the preceding commands to determine whether it should process the current command.

When the RESETRC command is processed, SMP/E resets the return codes for the preceding commands to 0. This allows the next command after the RESETRC command to be processed, regardless of whether preceding commands succeeded.







---

## Chapter 20. The RESTORE Command

At times, you may want to remove a SYSMOD that has been applied to the target libraries. For example, perhaps you installed a fix for a problem and got unexpected results. If you have not yet accepted the SYSMOD into the distribution libraries, you can use the RESTORE command to remove it from the target libraries.

The RESTORE command replaces the affected elements in the target libraries with the unchanged versions from the distribution libraries. (As a result, once you have accepted a SYSMOD into the distribution libraries, you cannot use RESTORE to remove it from the target libraries.)

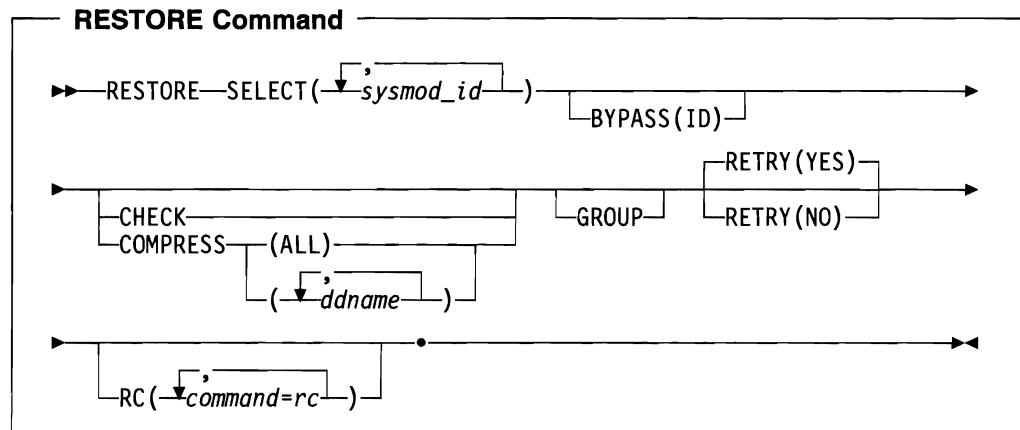
---

### Zones for SET BOUNDARY

For the RESTORE command, the SET BOUNDARY command must specify the target zone from which the SYSMODs should be removed. SMP/E uses the RELATED field in the TARGETZONE entry to determine which distribution zone describes the elements to replace the restored elements.

---

### Syntax



### Operands

#### BYPASS(ID)

indicates that SMP/E should ignore any errors it detects when checking RMID and UMIDs for element entries in the target zone or distribution zone.

#### CHECK

indicates that SMP/E should not actually update any libraries. Instead, it should just do the following:

- Test for errors other than those that can occur when the libraries are actually updated
- Report on which libraries are affected
- Report on any SYSMOD that would be regressed

### COMPRESS

indicates which target libraries should be compressed. SMP/E does **not** compress any libraries that are actually paths in a hierarchical file system.

- If you specify **ALL**, any libraries containing elements that will be updated by this RESTORE command are compressed.
- If you specify particular ddnames, those libraries are compressed regardless of whether they will be updated.

#### Notes:

1. **COMPRESS** can also be specified as **C**.
2. If you specify **COMPRESS** and **CHECK**, **COMPRESS** is ignored, because SMP/E does not update any data sets for **CHECK**.

### GROUP

indicates that if SMP/E determines that additional SYSMODs should be restored, other than those specified in the SELECT list, SMP/E should automatically include them.

For example, assume you have applied a function and service for that function. When you select the function and specify the **GROUP** operand, SMP/E also tries to restore the service that was applied for that function.

Likewise, assume you have applied two PTFs, and one defines the other as the prerequisite. When you select the prerequisite and specify the **GROUP** operand, SMP/E also tries to restore the other PTF. On the other hand, if you select the SYSMOD that specifies the prerequisite, SMP/E restores that particular SYSMOD **only** if the prerequisite has been accepted.

However, assume you have installed two PTFs that affect the same element but that do not define any relationship to each other. If you select one of the PTFs and specify the **GROUP** operand, SMP/E does **not** try to restore the other PTF. You have to specify both PTFs on the **SELECT** operand.

**Note:** **GROUP** can also be specified as **G**.

### RC

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the RESTORE command.

Before SMP/E processes the RESTORE command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the **RC** operand. If so, SMP/E can process the RESTORE command. Otherwise, the RESTORE command fails. For more information about the **RC** operand, see Appendix D.

#### Notes:

1. The **RC** operand must be the **last** operand specified on the command.
2. If you do specify the **RC** operand, return codes for commands not specified do not affect processing for the RESTORE command. Therefore, if you use the **RC** operand, you must specify every command whose return code you want SMP/E to check.

**RETRY**

indicates whether SMP/E should try to recover from out-of-space errors for utilities it calls.

**YES**

indicates that SMP/E should try to recover and retry the utility if a RETRYDDN list is available in the OPTIONS entry that is in effect. RETRY(YES) is the default.

If retry processing does not reclaim sufficient space and input to the utility was batched (copy or link-edit utility only), SMP/E debatches the input and retries the utility for each member separately. If this final attempt fails, the resulting x37 abend is treated as an unacceptable utility return code. In this case, processing continues for SYSMODs containing eligible updates to other libraries, but processing fails for SYSMODs containing unprocessed elements for the out-of-space library (and it fails for any SYSMODs that are dependent on the failed SYSMODs). For guidance on setting up the desired retry processing, see the *SMP/E R8.1 User's Guide*. For more information about OPTIONS entries, see "OPTIONS Entry (Global Zone)" on page 741.

If there is no RETRYDDN list, SMP/E does not try to recover from out-of-space errors, even if RETRY(YES) is specified.

**NO**

indicates that SMP/E should not try to recover from the error.

**SELECT**

specifies one or more SYSMODs that should be restored.

If you use **GROUP** along with **SELECT**, make sure to specify the lowest level of service you want restored. For example, if you want to restore PTF1 and PTF2, and PTF1 is a prerequisite for PTF2, specify PTF1 on the SELECT operand.

**Notes:**

1. SELECT is required for RESTORE. This is the only means of specifying which SYSMODs are eligible to be restored.
2. SELECT can also be specified as S.

---

**Data Sets Used**

The following data sets may be needed to run the RESTORE command. They can be defined by DD statements or, normally, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMPCTL	SMPPTS	SMPWRK2	SYSUT3
SMPCSI	SMPRPT	SMPWRK3	SYSUT4
SMPLOG	SMPSCDS	SMPWRK4	Distribution library
SMPLOGA	SMPSNAP	SYSLIB	Target library
SMPPLTS	SMPSTS	SYSPRINT	Link library
SMPMTS	SMPTLIB	SYSUT1	Text library
SMPOUT	SMPWRK1	SYSUT2	zone

## Notes:

1. The SMPLTS data set is required only if a load module having a CALLLIBS subentry list is being created, updated, deleted, or renamed.
2. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

---

## Usage Notes

- Certain conditions can cause SYSMODs to be considered ineligible for RESTORE processing. These conditions cause SMP/E to terminate processing of the ineligible SYSMODs and issue messages to inform you of the error conditions.

The following conditions cause SMP/E to consider a SYSMOD as ineligible for RESTORE processing:

- An element being restored has a MODID in the element entry on the distribution zone that does not have a corresponding SYSMOD entry on the target zone. This condition can occur if a SYSMOD has been accepted without being applied and, as a result, the distribution library is at a higher function or service level than the target system library.
- The service level of an element being restored is the same in the target library as it is in the distribution library. This condition can occur if a SYSMOD is both applied and accepted.
- A SYSMOD that should have been selected for RESTORE processing was not specified in the SELECT operand list. This condition can occur if one of the SYSMODs specified in the list is part of a RESTORE group that is not fully specified.
- The service level of an element in the distribution library is not the correct one. This can occur if several modifications to the same element are applied at different points in time, without being accepted, and the later modifications are the ones that are selected for RESTORE processing.

Consider the following example. The distribution zone shows that an element was last replaced on the distribution libraries by PTF UZ00001, but the related target zone indicates that the last replacement to the element on the system was by PTF UZ00004. The element was also modified on the system by PTFs UZ00002 and UZ00003. Figure 25 on page 337 shows the SYSMODs on the related target zone and distribution zone in service order.

If you specified the following, PTFs UZ00002 and UZ00003 would not be considered part of the RESTORE processing group because they are not dependent on PTF UZ00004.

```
SET      BDY(TGT1)      /* Set to target zone.*/.
RESTORE  S(UZ00004)    /*
GROUP    GROUP        /* */.
```

To correct the error, specify:

```
SET      BDY(TGT1)      /* Set to target zone.*/.
RESTORE S(UZ00002)     /*          */
GROUP   GROUP          /*          */.
```

When this condition is detected, SMP/E issues messages to inform you of the SYSMODs that must be restored along with the specified SYSMOD or accepted before that SYSMOD is restored.

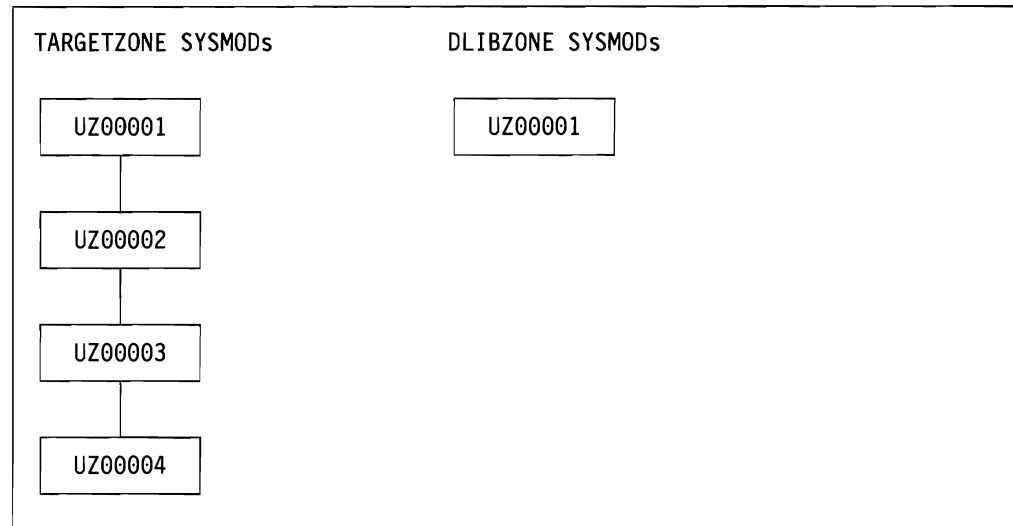


Figure 25. SYSMODs for RESTORE Example

- The ineligibility of a member of a RESTORE group terminates processing for the entire group. This can occur both in GROUP and SELECT mode.
- A function SYSMOD containing a ++VER DELETE MCS cannot be restored if any of the specified SYSMODs were actually deleted when the function was applied. (Such a function is eligible for RESTORE processing if none of the specified SYSMODs had ever been applied, and were, therefore, not deleted when the function was installed.)

Function SYSMODs containing a ++DELETE statement for a load module are not eligible for RESTORE processing.

If a function SYSMOD is terminated for any of the above conditions, the RESTORE function is also terminated.

- You can avoid certain error conditions that would terminate a SYSMOD by specifying the BYPASS(ID) operand on the RESTORE command. Then, error conditions in the ID validation checking do not cause SYSMOD termination, but are treated as warnings.

The first two conditions described earlier in the first special consideration (SYSMOD ineligibility) can be bypassed by using this option. However, in the first case, the distribution library contains a version of the element that is probably functionally superior to the version being removed. This can cause the executable code in the target system library to be inoperable. In addition, SMP/E updates the element entry on the target zone to reflect the UMID and RMID subentry contents from the element entry on the distribution zone. In this case, the SYSMOD entry might not exist on the target zone, because the BYPASS(APPLYCHECK) operand was probably used on the ACCEPT

command; thus, the SYSMOD was never applied to the target system. You should avoid using the BYPASS(ID) option unless it is absolutely necessary.

- Utility failures can cause the RESTORE command to fail. For details on handling x37 abends, see the description of the RETRY operand under “Operands” on page 333.
- SYSMOD entries on the target zone have the ERROR and RESTORE status indicators set on before the target system libraries are updated. If processing fails during the updating, these indicators remain on and the updating for these entries is not completed. After you determine the cause of the termination, you can process these SYSMODs again by specifying them as operand values of the SELECT operand on the RESTORE command.
- RESTORE processing does not replace the nucleus with the saved copy, but relinks it using the last version of modules accepted on the DLIBs. The saved nucleus is available only to provide an alternative nucleus for IPL should an applied SYSMOD damage IEANUC01.
- When a selected SYSMOD contains an element that was deleted from the system by that SYSMOD, RESTORE processing reintroduces that element into the target system using information saved in the SMPSCDS BACKUP entries.
- If you do not use SMP/E to recover after a failure, and choose the option of restoring your system and the distribution libraries via system and DLIB RESTORE tapes, you must ensure that the SMPPTS, SMPCSI, SMPSCDS, SMPMTS, and SMPSTS data sets are also restored to their previous levels.
- The exception SYSMOD data stored in the global zone SYSMOD entry is not purged when the SYSMOD is restored. If NOREJECT is not set in the OPTIONS entry that is in effect, the global zone SYSMOD entry is purged of all information except the exception SYSMOD data. (Having NOREJECT set off is the default.)

---

## Output

The following reports may be produced during RESTORE processing:

- Causer SYSMOD Summary report
- Cross-Zone Summary report
- File Allocation report
- Element Summary report
- MOVE/RENAME/DELETE report
- SYSMOD Status report

See Chapter 31 for descriptions of these reports.

## Examples

In each of these examples, the following set of SYSMODs has been received:

```

++PTF(UZ00001)          /* 1st PTF in chain 1.    */.
++VER(Z038) FMID(FXY1040) /* No prerequisites.     */.
++MOD(XYMOD01)          /*                          */.

++PTF(UZ00002)          /* 2nd PTF in chain 1.    */.
++VER(Z038) FMID(FXY1040) /*                          */.
PRE(UZ00001)           /* Previous PTF.         */.
++MOD(XYMOD01)          /*                          */.

++PTF(UZ00003)          /* 3rd PTF in chain 1.    */.
++VER(Z038) FMID(FXY1040) /*                          */.
PRE(UZ00002)           /* Previous PTF.         */.
++MOD(XYMOD01)          /*                          */.

++PTF(UZ00010)          /* 1st PTF in chain 2.    */.
++VER(Z038) FMID(FXY1040) /*                          */.
PRE(UZ00001)           /* Logical requisite.     */.
++MOD(XYMOD02)          /*                          */.

```

**Note:** For these examples, assume (1) all modules are present in the target zone and distribution zone, with the result that the DISTLIB operand is not required; and (2) the actual module replacement follows the ++MOD statement.

The following examples are provided to help you use the RESTORE command:

- “Example 1: Restoring a Single SYSMOD”
- “Example 2: Restoring Multiple PTFs to Remove One PTF” on page 340
- “Example 3: Restoring PTFs Using the GROUP Operand” on page 340

### Example 1: Restoring a Single SYSMOD

Assume you have applied only PTF UZ00001, that an error was detected during testing, and that you want to remove the PTF from your system. The following RESTORE command can be used:

```

SET      BDY(TGT1)          /* Set to target zone.    */.
RESTORE S(UZ00001)          /* Restore 1 PTF.        */.

```

If you want to clean up all of SMP/E's records for this PTF (the global zone and the SMPPTS data set), you can use the REJECT command after RESTORE processing is complete:

```

SET      BDY(GLOBAL)        /* Set to global zone.    */.
REJECT  S(UZ00001)          /* Reject 1 PTF.         */.

```

## Example 2: Restoring Multiple PTFs to Remove One PTF

Assume you have applied all PTFs UZ00001, UZ00002, and UZ00003 to your system, and that during testing an error is found in module XYMOD01. Because the current service level of that module is UZ00003, you want to restore that PTF from the system.

You now have two choices:

1. Restore PTF UZ00001, UZ00002, UZ00003, and then reapply UZ00001 and UZ00002 as follows:

```
SET      BDY(TGT1)           /* Set to target zone.  */.
RESTORE  S(UZ00001,         /* Restore all 3 PTFs.  */
          UZ00002,         /*                       */.
          UZ00003)        /*                       */.
APPLY    S(UZ00001         /* Then re-apply the two */
          UZ00002)        /* that may be ok.     */.
```

2. Accept PTFs UZ00001 and UZ00002, if you are sure that they have no errors, then restore UZ00003 as follows:

```
SET      BDY(DLIB1)         /* Set to DLIB zone.    */.
ACCEPT   S(UZ00001,        /* Accept two good PTFs.*/
          UZ00002)        /*                       */.
SET      BDY(TGT1)         /* Set to target zone.  */.
RESTORE  S(UZ00003)        /* Restore the 1 bad PTF.*/.
```

The end result in both cases is that module XYMOD01 from PTF UZ00002 is in the target libraries.

## Example 3: Restoring PTFs Using the GROUP Operand

In Example 2, when you wanted to restore the three PTFs, you specified all three in the select list. In a simple case like this, that was very easy; in practice, however, many PTFs are related to one another, and it may not be easy to determine which PTFs must be restored in order to remove the bad one. The GROUP operand can be used to assist in determining this. The following commands can be run to determine which PTFs must be restored to restore UZ00003:

```
SET      BDY(TGT1)           /* Set to target zone.  */.
RESTORE  S(UZ00003)         /* Restore this one PTF */
          GROUP             /* plus any related PTFs, */
          CHECK             /* in check mode this time.*/.
```

After running these commands, the various SMP/E reports can be used to determine that PTFs UZ00001, UZ00002, and UZ00003 should be restored. You can then determine the correct action: restore all, or accept some and then restore.



---

## Processing

This section describes the following:

- Selecting SYSMODs
- Installing elements
- Recording After completion
- Cross-zone processing
- Global zone SYSMOD entries

## SYSMOD Selection

This section describes how SMP/E selects SYSMODs.

### Operands Related to SYSMOD Selection

You can use the following operands to tell SMP/E which SYSMODs are to be restored:

- SELECT
- GROUP

SELECT tells SMP/E which particular SYSMODs are to be restored. GROUP, while having an effect on SYSMOD selection, does not directly indicate which SYSMODs are affected.

### Candidate Selection

When the RESTORE command is encountered, SMP/E looks at each SYSMOD specified in the SELECT list to make sure the following conditions hold:

- The SYSMOD has been applied to the target zone specified in the previous SET command.
- The SYSMOD has not been accepted to the distribution zone specified in the RELATED field of the TARGETZONE entry for the target zone specified in the previous SET command.

If any SYSMODs are found that do not meet both of these conditions, error messages are written to the SMPLOG and SMPOUT, and those SYSMODs are not considered eligible to be restored.

Once the SYSMODs specified in the SELECT list have been checked for initial eligibility, SMP/E checks to see whether any other SYSMODs are affected. There are two ways other SYSMODs can be affected:

- SYSMODs can be related to one another through the PRE, REQ, FMID, and SUP operands of their ++VER statement or the REQ operand of the ++IF statement.

For each candidate SYSMOD, SMP/E checks to see whether dependencies exist between it and any other SYSMOD not yet accepted. (That is, does any PRE, REQ, IFREQ, or FMID relationship exist between the candidate SYSMOD and these other SYSMODs?) If so, that SYSMOD must also be restored. This is true because of the stated dependency on either the functional or service dependency of the SYSMODs.

- SYSMODs can be related to one another, because they have elements in common.

For each candidate SYSMOD, SMP/E checks to see if any other SYSMOD, not yet accepted, has modules in common with the candidate SYSMOD. If so, that SYSMOD must also be restored. This is true because of the method used to replace the elements on the system libraries. The version of the element in the distribution library is used as the backup. Thus, all SYSMODs that have replaced or modified the elements since the distribution library version was accepted must also be removed.

Processing of these related SYSMODs depends on whether the GROUP operand was specified:

- If the GROUP operand was specified, each of the related SYSMODs, as identified above, are included as candidates for RESTORE. SMP/E then performs the same checking on these new candidates as on the original set. This process continues until no additional SYSMODs are added.
- If the GROUP operand was not specified, SMP/E issues an error message to SMP/OUT and SMP/LOG indicating which of the SYSMODs specified in the SELECT list cannot be restored. This information can also be found in the RESTORE SYSMOD Status report.

## Element Installation

Once the proper SYSMODs have been selected, SMP/E moves the version of the element in the distribution libraries to the proper places in the system libraries. Processing for each type of elements is described in subsequent sections.

### Inline JCLIN

If a SYSMOD that had inline JCLIN is restored, SMP/E attempts to restore the target zone entries affected by the JCLIN to the state they were in before the SYSMOD was applied. This is done by accessing the BACKUP entry for such SYSMODs. For each BACKUP entry, SMP/E checks the corresponding target zone entry to ensure that the last modification (LASTUPD subentry) to the target zone entry was for the SYSMOD being restored. If it was, the entry is replaced from the BACKUP entry. If it was not, SMP/E issues messages to indicate that the SYSMOD was not restored, and RESTORE processing stops for that SYSMOD. This condition can occur if you used UCLIN or JCLIN to update an entry after you applied the SYSMOD being restored, or if a subsequent SYSMOD was applied that updated the entry but did not have a dependency relationship with the SYSMOD being restored. The latter should occur only for LMOD entries.

**Note:** RESTORE processing is limited for a SYSMOD using the CHANGE statement in inline JCLIN. When that SYSMOD is restored, the backup copy of the LMOD entry (which does not have the updates from the CHANGE statement) replaces the target zone LMOD entry, and the information from the CHANGE statement is lost. Module names that were changed by the inline JCLIN remain in the load module under their changed names.

As each entry is completed, SMP/E deletes the BACKUP entry. When all BACKUP entries have been processed, SMP/E deletes the related SYSMOD entry. This processing is done before the target system libraries are updated.

JCLIN processing occurs in the reverse order of application; that is, the latest update is restored first, the earliest one last. The order is determined by the dependency relationships of the SYSMODs being restored.

### Deleted Elements

If a SYSMOD being restored had element MCSs with the DELETE operand, SMP/E attempts to bring back the target zone element entries that were deleted when the SYSMOD was applied. This is done by using the BACKUP entries for the SYSMOD. For each BACKUP element entry, SMP/E checks whether there is a corresponding entry in the target zone.

- If there is no target zone entry for the element, SMP/E copies the BACKUP element entry into the target zone.
- If there is a target zone entry for the element, SMP/E issues a message to indicate that the entry has not been replaced with the BACKUP entry, and RESTORE processing continues.

This can happen if you used UCLIN or JCLIN to recreate an entry after you applied the SYSMOD being restored, or if a subsequent SYSMOD was applied that recreated the entry but did not define a relationship with the SYSMOD being restored.

As SMP/E completes processing for each element, it deletes the corresponding BACKUP entry. When all BACKUP entries for the SYSMOD have been processed, SMP/E deletes the related SYSMOD entry. It then updates the target libraries using the procedure described in the following sections.

### Compressing the Target Libraries

You can use the COMPRESS operand of the RESTORE command to have SMP/E compress the target libraries before restoring SYSMODs. There are two methods of specifying which libraries are to be compressed:

- You can specify selected libraries in the COMPRESS operand (including libraries that may not be affected by the RESTORE command).
- You can specify **ALL** in the COMPRESS operand; only libraries in which elements will be restored by this RESTORE command are then eligible for compression.

**Note:** Target libraries residing in a hierarchical file system are not compressed.

Once the libraries have been determined, actual processing is dependent on the library type.

- For **source** libraries, any source to be replaced (not updated) by one of the SYSMODs being restored is deleted from the library.
- For **macro** libraries, no macros are deleted, because the SYSMOD replacing the macro might fail, and the failure could lead in turn to the failure of other SYSMODs causing assemblies that use the macro.
- For **data element** libraries, any data element that is to be replaced by one of the SYSMODs being restored is deleted from the library.
- For **load** libraries, SMP/E determines which load modules within the library are composed only of modules that were copied during system generation and are being replaced by the SYSMODs being restored. Such load modules are deleted from the library.

SMP/E then calls the copy utility to perform the actual compress operation.

**Note:** To reclaim as much space as possible before restoring the SYSMODs, members are deleted before the library is compressed. SMP/E processes one library at a time, first deleting members, then compressing the library.

### Data Elements and HFS Elements

Data elements and HFS elements are restored before other types of elements. They are copied from the distribution libraries to the target libraries using the copy utility for data elements and the HFS copy utility for HFS elements.

### Macros

Macros existing in a target system library (that is, their SYSLIB subentry is non-blank) are simply copied from the distribution library into the appropriate target library. If no version of the macro is found in the distribution library, the macro is deleted from the target library.

For a macro that does not reside in any target library, SMP/E simply removes the current version of that macro from the SMPMTS.

### Source

The processing of source code is exactly the same as the processing of macros, except that the SMPSTS is used rather than the SMPMTS.

### Assemblies

RESTORE processing for assemblies is done in exactly the same way as during APPLY processing. For the detailed description, see "Assemblies" on page 95.

### Modules

RESTORE processing for modules is essentially the same as APPLY processing, except that the source for the module replacement is the module distribution library rather than the selected version from a SYSMOD. For the detailed description, see "Module Replacements" on page 97.

### Notes:

1. A superzap (that is, ++ZAP) is also considered a module, because the whole module is replaced from the distribution libraries.
2. If the SYSMOD being restored added an existing module to an existing load module, RESTORE processing does not remove that module from the target libraries or from the load module.

Typically, in such cases, the SYSMOD updated modules or added new modules to call the existing module in the load module. So, when the SYSMOD is restored, the new modules are deleted from the target libraries, the DLIB versions of the updated modules are restored to the target libraries, and the load module is relinked to remove the new modules and to pick up the restored modules. As a result, although the existing module is still physically in the load module, it has been logically removed, because the modules that called it are gone.

3. If MODDEL subentries were added to LMOD entries to indicate that a module was deleted during APPLY processing, the MODDEL subentries are deleted from the LMOD entries during RESTORE processing.

4. SMP/E checks whether load modules to be updated have XZMOD subentries with the same name as a module selected to update the load module. If so, SYSMOD processing stops.

### Load Modules Created by the SYSMOD Being Restored

Normally, if a load module was originally created by the SYSMOD being restored, SMP/E deletes the load module and the associated LMOD entry. If such a load module contains cross-zone modules, however, SMP/E does not delete the load module or the LMOD entry. Instead, it invokes the linkage-editor to remove the modules that are defined in the same zone as the load module (leaving a *stub* load module in the target library).

### Load Modules with a SYSLIB Allocation

If RESTORE processing replaces a load module having a SYSLIB allocation with a version of the load module that does not have one, the base version of the load module is deleted from the SMPLTS data set. (In this case, the load module's LMOD entry in the target zone contains a CALLLIBS subentry list; that entry is replaced by an LMOD entry from the SMPSCDS that does not contain a CALLLIBS subentry list.) As a result, the executable version of the load module in its target libraries may contain modules that were included by the automatic library call mechanism when the load module was link-edited during APPLY processing of the SYSMOD now being restored. If there are still external references to these modules, they may continue to function in the load module; otherwise, they become inactive code in the load module.

### Deleted Load Modules

SMP/E cannot restore a load module that was deleted by the ++DELETE statement.

### Moved Elements and Load Modules

If a macro, a module, a source, or a load module was moved by a ++MOVE statement, SMP/E returns it to its original library and deletes it from the one it was moved to. If a ++MOVE statement moved an element from one distribution library to another, the DISTLIB ddname in the target zone element entry is restored to its value before the move.

### Renamed Load Modules

If a load module was renamed by the ++RENAME statement, SMP/E restores its original name.

**Note:** If a SYSMOD being restored contained a ++RENAME statement for a load module containing cross-zone modules, SMP/E checks whether those zones indicate that cross-zone updates should be done automatically. (This is done if you specify the AUTOMATIC option.) If so, the cross-zone MOD entries are updated during cross-zone processing. For more information, see "TARGETZONE Entry (Target Zone)" on page 782.

### Recording After Completion

Results of processing are recorded in the following entries:

- “Target Zone Element Entries”
- “SMPSCDS BACKUP Entries”
- “Target Zone SYSMOD Entries”

#### Target Zone Element Entries

The various function and service level fields (FMID, RMID, and UMID) in the target zone entries are modified to be the same as they are in the corresponding distribution zone entry.

#### SMPSCDS BACKUP Entries

For each SYSMOD successfully restored that had inline JCLIN, the corresponding SMPSCDS BACKUP entry is deleted.

#### Target Zone SYSMOD Entries

**Superseded SYSMODs:** All SYSMOD entries that are superseded by SYSMODs being restored have the SUPBY subentries for those SYSMODs deleted. If all the SUPBY subentries for a superseded SYSMOD are deleted, the SYSMOD entry itself is deleted. As a result of restoring a SYSMOD that superseded a previously applied SYSMOD, target zone entries that might have been ignored during APPLY processing may now be applicable. This condition is not acted upon by RESTORE processing. Therefore, subsequent apply processing may request requisite SYSMODs that are now applicable because of previously applied function SYSMODs.

**SYSMOD Entry:** When a SYSMOD is successfully restored, the SYSMOD entry is deleted from the target zone.

### Cross-Zone Processing

If entries for modules or load modules that have been successfully restored contain cross-zone subentries, and if the associated cross-zones can be automatically updated, SMP/E does cross-zone processing.

First, SMP/E obtains access to the CSIs containing the cross-zones. It then checks each cross-zone to make sure it contains DDDEF entries for the target libraries needed for link-edit processing.

For each restored module that is part of a cross-zone load module, SMP/E checks the cross-zone LMOD entries to make sure the set-to zone originally supplied the modules to be processed. If so, SMP/E does the following:

- If the module was replaced by a distribution zone copy of the module, SMP/E schedules link-edit processing to include the replacement module.
- If the module was deleted, SMP/E schedules link-edit processing to delete that module.

**Note:** If a cross-zone LMOD entry to be processed consists only of cross-zone subentries, no processing is done for that load module. The load module no longer really exists.

For each cross-zone module contained in a renamed LMOD that was restored in the set-to zone, SMP/E changes the XZLMOD subentry to reflect the old LMOD name.

The Cross-Zone Summary report provides a summary of all the cross-zone work done, except for cross-zone work caused by renamed LMODs. This is summarized in the MOVE/RENAME/DELETE report.

## Global Zone SYSMOD Entries

When a SYSMOD is successfully restored and the NOREJECT indicator in the OPTIONS entry in use is off, the global zone SYSMOD entry and the SMPPTS MCS entries are deleted along with any SMPTLIB data sets associated with that SYSMOD.

If the SYSMOD being restored has any external exception SYSMOD data (that is, ++HOLD data) associated with it, that information is not deleted when the SYSMOD entry is deleted. This allows you to restore a SYSMOD and then modify it and receive it again without having to rereceive the exception data associated with it. If the SYSMOD itself contained a ++HOLD statement, it is considered part of the SYSMOD and is, therefore, deleted along with the SYSMOD.

When a SYSMOD is successfully restored and the NOREJECT indicator in the OPTIONS entry in use is on, the APPID subentry matching the target zone name is deleted, indicating that the SYSMOD is no longer applied to that zone.

## Zone and Data Set Sharing Considerations

The following identifies the phases of RESTORE processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

### 2. RESTORE processing

Target zone	—	Update with exclusive enqueue.
DLIB zone	—	Update with shared enqueue.

### 3. Global zone update

Global zone	—	Update with exclusive enqueue.
SMPPTS	—	Update with exclusive enqueue.
Target zone	—	Update with exclusive enqueue.

### 4. Cross-zone processing

Distribution zone	—	Read with shared enqueue.
Cross-zones	—	Read with shared enqueue.
Distribution zone	—	Update with exclusive enqueue.
Cross-zones	—	Update with exclusive enqueue.
Global zone	—	Read with no enqueue.

### 5. Termination

All resources are freed.





---

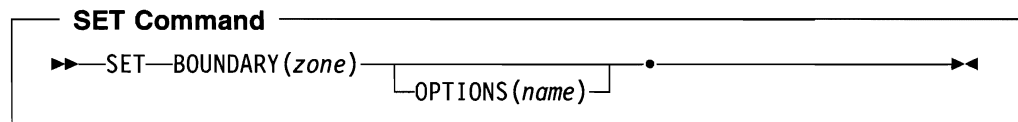
## Chapter 21. The SET Command

Most SMP/E commands update certain zones. For example, the ACCEPT command updates a distribution zone, and the APPLY command updates a target zone. To specify which zone should be updated by a given command, you must use the SET command. The zone is identified on the BOUNDARY operand, which indicates that all subsequent commands, up to the next SET command, should be processed for the specified zone.

The SET command can also be used to request a particular set of predefined operating options. This is done with the OPTIONS operand. The OPTIONS operand specifies the global zone OPTIONS entry containing the processing options to be used for all subsequent commands, until the next SET command.

---

### Syntax



### Operands

#### BOUNDARY

specifies which zone should be updated by the commands following the SET command.

#### Notes:

1. BOUNDARY is a required operand.
2. BOUNDARY can also be specified as **BDY**.

#### OPTIONS

specifies an OPTIONS entry that should be used for the commands following the SET command. If an OPTIONS entry is specified, it overrides the one specified in the zone definition.

#### Notes:

1. The specified OPTIONS is used only to process the zone specified on the BOUNDARY operand.
2. For cross-zone processing, SMP/E uses the OPTIONS entry specified in the TARGETZONE entry for the cross-zone. If no OPTIONS entry is defined for the cross-zone, SMP/E uses default values when doing work to the cross-zone.

### Data Sets Used

The following data sets may be needed to run the SET command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOG	SMPOUT	<i>zone</i>
SMP_CSI	SMPLOGA	SMPSNAP	

#### Notes:

1. If SMP/E does not find the SMPLOG DD statement when parsing the SET command, SMP/E buffers all messages until after the specified zone has been determined. At that time, SMP/E accesses that zone to try to dynamically allocate the SMPLOG DD statement.
2. If SMP/E does not find the SMPOUT DD statement when parsing the SET command, SMP/E buffers all messages until after the specified zone has been determined. At that time SMP/E accesses that zone to try to dynamically allocate the SMPOUT DD statement.
3. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

### Usage Notes

SMP/E uses the SET command to control dynamic allocation. When SMP/E needs to dynamically allocate a data set, it allocates that data set once per zone. That data set remains allocated until the next SET command is processed. If SMP/E fails to dynamically allocate a data set, it keeps a record of that unsuccessful attempt and does not try to reallocate the data set. When SMP/E processes the next SET command, it frees all dynamically allocated data sets and erases the records of allocation attempts that failed. This has certain benefits:

- Each zone can use different definitions for the same data set.
- Performance is improved, because SMP/E does not need to dynamically allocate and free each data set every time it is needed.

For more information about dynamic allocation, see Appendix C.

### Examples

The following examples are provided to help you use the SET command:

- “Example 1: Receiving SYSMODs into the SMPPTS Data Set” on page 351
- “Example 2: Applying SYSMODs to the Target Libraries” on page 351
- “Example 3: Accepting SYSMODs to the Distribution Libraries” on page 351
- “Example 4: Processing Multiple Commands in One Invocation of SMP/E” on page 352

- “Example 5: Changing Which OPTIONS Entry Is Used” on page 352
- “Example 6: Resolving Errors in Dynamic Allocation” on page 352

### Example 1: Receiving SYSMODs into the SMPPTS Data Set

To receive SYSMODs into the SMPPTS data set, SMP/E must be directed to process the global zone. Suppose you want to receive PTFs from an ESO tape containing service level 9301 into the SMPPTS. To do this, specify the following set of commands:

```
SET      BDY(GLOBAL)      /* Process global zone.  */.
RECEIVE SYSMODS          /* Receive SYSMODs.     */.
LIST     SYSMODS         /* List SYSMOD entry    */.
          MCS            /* and MCS               */.
          SOURCEID(PUT9301) /* for SYSMODS received. */.
```

This causes all applicable SYSMODs to be received and to be assigned the source ID, specified in the ESO (in this case, 9301). The LIST command causes SMP/E to list the MCS entries for all the SYSMODs just received.

### Example 2: Applying SYSMODs to the Target Libraries

After receiving a set of SYSMODs, the next step is to apply them to the target libraries. To do this, the SET command must specify the target zone associated with those libraries. In this example, the SYSMODs are being installed into a target zone named MVSTST1 that represents a set of test libraries. The following commands are required to apply a SYSMOD:

```
SET      BDY(MVSTST1)    /* Set to process MVSTST1. */.
APPLY    SOURCEID(PUT9301) /* Apply service level     */.
          GROUP          /* with group to pick up   */.
                      /* previous resolved       */.
                      /* exception SYSMODs.     */.
```

The result is that all PTFs that were received and assigned a source ID of PUT9301, and that are applicable to the functions in the MVSTST1 target system, are applied.

**Note:** The GROUP operand automatically includes requisites for the PTFs with the indicated source ID.

### Example 3: Accepting SYSMODs to the Distribution Libraries

After applying a set of SYSMODs, the final step is to accept them into the distribution libraries. To do this, the SET command must specify the distribution zone associated with those libraries. In this example, the SYSMODs will be installed into a distribution zone named MVSDLB1. The following commands are required to accept a SYSMOD:

```
SET      BDY(MVSDLB1)    /* Set to process MVSDLB1. */.
ACCEPT   SOURCEID(PUT9301) /* Accept service level.   */.
```

The result is that SMP/E accepts all PTFs that were received and assigned a source ID of PUT9301, that have been applied, and that are applicable to the functions in the MVSDLB1 distribution zone.

## Example 4: Processing Multiple Commands in One Invocation of SMP/E

The preceding set of examples showed how the SET command is used to define the scope of processing to SMP/E when only one operation is to be performed at a time. SMP/E makes it possible to perform all these operations during one invocation, if desired. The commands are as follows:

```

SET      BDY(GLOBAL)          /* Process global zone.   */.
RECEIVE  SYSMODS              /* Receive SYSMODs.       */.
          SOURCEID(PUT9301)   /* Assign SOURCEID.      */.
LIST     SYSMODS              /* List SYSMOD entry     */.
          MCS(PUT9301)        /* and MCS                */.
          SOURCEID(PUT9301)   /* for SYSMODs received. */.
SET      BDY(MVSTST1)        /* Set to process MVSTST1.*/.
APPLY    SOURCEID(PUT9301)   /* Apply service level.   */.
SET      BDY(MVSDLB1)        /* Set to process MVSDLB1.*/.
ACCEPT   SOURCEID(PUT9301)   /* Accept service level.  */.

```

**Note:** In a job with multiple SET commands, if you use DDDEF entries that specify SYSOUT for SMP/E output (such as SMPOUT or SMPRPT), SMP/E produces multiple SYSOUT data sets. This can cause undesirable results; for example, the output may appear to be out of sequence from one SET command to the next. Therefore, when you run such a job, you may prefer to use DD statements, rather than DDDEF entries, for SMP/E output data sets.

## Example 5: Changing Which OPTIONS Entry Is Used

SMP/E allows you to define multiple OPTIONS entries in the global zone so that various processing options can be used as required. The global zone and each target zone and distribution zone identify the default OPTIONS entry to be used in processing that zone. At times, you may require that a different OPTIONS entry be used for the installation of a given product or PTF. Rather than change the name of the default OPTIONS entry in the zone definition, SMP/E allows you to override the default OPTIONS name on the SET command.

For example, suppose you want to use the default OPTIONS entry to install all the service PTFs in service level 9301, but PTF UR12345 must be installed using another OPTIONS entry (previously defined with the correct unique processing options for this PTF). You can use the following set of commands:

```

SET      BDY(MVSTST1)        /* Set to process MVSTST1.*/.
APPLY    SOURCEID(PUT9301)   /* Apply all PTFs        */.
          EXCLUDE(UR12345)   /* except UR12345.      */.
SET      BDY(MVSTST1)        /* Reset to change       */.
          OPTIONS(URPTFS)    /* OPTIONS entry used.   */.
APPLY    S(UR12345)         /* Now apply UR12345.   */.

```

## Example 6: Resolving Errors in Dynamic Allocation

During processing, SMP/E attempts to dynamically allocate a data set one time per zone. If the allocation fails, SMP/E remembers and uses the information if the data set is requested again. For this example, let us assume that you are calling SMP/E from a terminal and that you have entered the following:

```

SET      BDY(MVSDLB1)        /* Set to process MVSDLB1.*/.
ACCEPT   S(UR12345)         /* Accept UR12345.      */.

```

Also, assume PTF UZ12345 requires distribution library AMACLIB, but that no DD statement has been allocated and no DDDEF entry is present. SMP/E issues an error message indicating the AMACLIB could not be allocated because no DDDEF entry was found, and the accept of the PTF fails. You can correct the problem by entering the following commands:

```

RESETRC                /* Allows UCLIN to run after
                        accept failed.          */.
UCLIN                  /* Add AMACLIB DDDEF.      */.
  ADD    DDDEF(AMACLIB) /* Add DDDEF          */.
        DA(SYS1.AMACLIB) /* with data set     */.
        OLD              /* and disposition.  */.
ENDUCL                 /* End UCL changes.  */.
SET    BDY(MVSDLB1)    /* Set to process MVSDLB1.
                        Will also cause
                        allocation history for
                        AMACLIB to be deleted. */.
ACCEPT  S(UR12345)     /* Accept UR12345.    */.

```

If the SET command had not been specified after the UCLIN operation, SMP/E would have issued a message indicating that an earlier attempt to allocate AMACLIB had failed and that no allocation attempt was made. As a result, the ACCEPT would have failed again.

## Processing

When a SET command is encountered, SMP/E attempts to open the data set containing that zone. The data set to be opened is identified by looking in the global zone ZONEINDEX list.

- If no ZONEINDEX subentry exists, SMP/E reports an error condition.
- If a ZONEINDEX subentry exists, SMP/E checks to see if the data set specified for that zone is already open; if so, it does no further processing.
- If the data set containing the zone is not already open, SMP/E checks to see whether a DD statement has been provided (the ddname is equal to the zone name).
  - If a DD statement has been provided, the data set pointed to by the DD statement is opened.
  - If no DD statement has been provided, SMP/E attempts to dynamically allocate a DD statement using the zone name as the ddname and the data set specified in the ZONEINDEX as the data set name.

Processing then continues with the next command.

Some common errors that can occur during a SET command are:

- The specified zone cannot be found in the global zone ZONEINDEX. In this case, SMP/E checks the syntax of all subsequent commands, but does not process them because it cannot determine where to direct the processing. To fix this problem, define the zones and rerun the job.
- The specified zone cannot be found on the data set specified in the global zone ZONEINDEX or the data set pointed to by the DD statement for that zone. In this case, the only SMP/E command that can be executed is the UCLIN

command to define the zone definition entry. Other SMP/E commands fails because of insufficient information to process them.

---

### Zone and Data Set Sharing Considerations

The following identifies the phases of SET processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

#### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

**Note:** The type of zone that is accessed depends on the zone specified in the SET command.

#### 2. Global zone update

Global zone	—	Update with exclusive enqueue.
SMPPTS	—	Update with exclusive enqueue.
Target zone	—	Update with exclusive enqueue.
DLIB zone	—	Update with exclusive enqueue.

**Notes:**

- a. This phase is executed only if the zone type in the SET command was either a target zone or a distribution zone, and only if that target zone or distribution zone contained pending global zone updates from a previous APPLY, ACCEPT, or RESTORE command.
- b. Either the target zone or distribution zone is accessed, according to the zone type specified in the SET command.

#### 3. Termination

All resources are freed.

---

## Chapter 22. The UCLIN Command

With the UCLIN command you can add, delete, or replace entries in the following SMP/E data sets:

- SMPCSI
- SMPMTS
- SMPSCDS
- SMPSTS

**Note:** With the UCLIN command, you can make changes similar to those that can be made to other data sets with the IMASPZAP and IEHPROGM utilities. However, you cannot use a utility or an editor to change the information in the data sets listed above; you must use the UCLIN command.

UCLIN updates only entries in SMP/E data sets. It does nothing to any elements or load modules in any product libraries. You must ensure that the appropriate changes are made to the libraries.

Be sure you understand the relationships between the various entries before making any UCLIN changes. This helps ensure that any UCLIN changes you make are complete and consistent with one another. When SMP/E processes UCLIN, it checks only the specified entry. It does not check how the changes might affect other entries.

The following terms are used in this discussion of UCLIN processing:

**subentry:** A field within an entry. Each subentry has an associated type and value. An example of a single-value subentry is the PEMAX subentry in the OPTIONS entry.

**subentry list:** Multiple occurrences of the same subentry type in an entry, each with a different value. For example, the modules supplied by a PTF are saved as names in the MOD subentry list within the PTF's SYSMOD entry.

**subentry indicator:** A field in an entry that does not have a data value associated with it. An example of a subentry indicator is the APP indicator in a SYSMOD entry.

---

### Zones for SET BOUNDARY

For the UCLIN command, the SET BOUNDARY command must specify either the zone whose entries are to be changed, or the zone containing the DDDEF entry for the data set that is to be changed.

---

### UCLIN and ENDUCL Syntax

Three types of statements are needed for UCLIN processing:

1. The **UCLIN** command indicates the start of UCL processing.
2. **UCL statements** follow the UCLIN command and describe the changes for a specific entry. There are three types of UCL statements: ADD, DEL, and REP.

These statements can add, delete, or replace entries or subentries in the entries.

**ADD** is used to add the following:

- A new entry
- A new subentry to an existing entry
- A new subentry list to an existing entry
- A new subentry list value to an existing subentry list in an existing entry
- A new subentry indicator to an existing entry

**DEL** is used to delete the following:

- An entire entry
- A subentry
- A complete subentry list
- A value from a subentry list
- A subentry indicator

**REP** is used to replace the following:

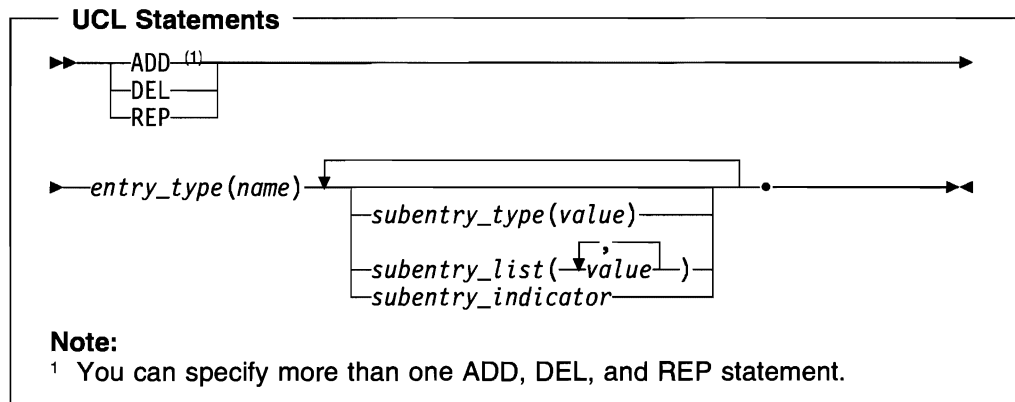
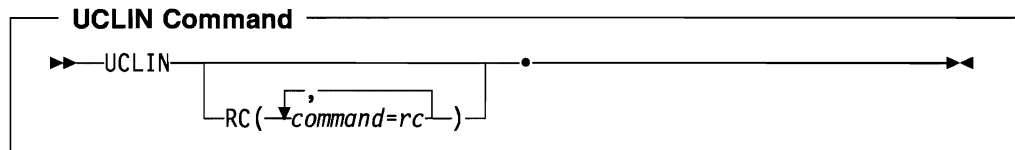
- A subentry in an existing entry
- A subentry list in an existing entry
- A subentry indicator in an existing entry

**Note:** Do **not** use the REP statement to replace an individual value in a subentry list. If the entry already contains the specified subentry list—for example, **F MID(ABC1234,DEF5678)**—SMP/E replaces **all** the current values with the new value specified on the REP statement.

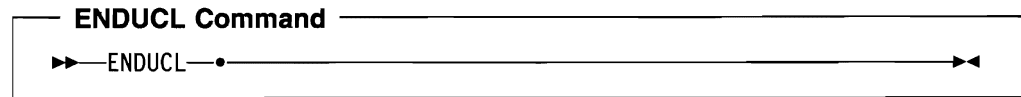
Many UCL statements can follow a single UCLIN command. “UCL Statement Syntax” on page 357 describes the syntax of specific UCL statements for each entry type.

3. The **ENDUCL** command indicates the end of the UCL statements and the end of UCLIN processing.

This is the general syntax for these statements:







## Operands

### RC

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the UCLIN command.

Before SMP/E processes the UCLIN command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the UCLIN command. Otherwise, the UCLIN command fails. For more information about the RC operand, see Appendix D.

#### Notes:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the UCLIN command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

#### *entry-type*

specifies the entry type to be updated. "UCL Statement Syntax" shows the entry types that can be specified. These entries are described in more detail in Chapter 35.

#### *name*

specifies the name of the entry to be updated.

#### *subentry-type*

specifies the subentry type to be updated. "UCL Statement Syntax" shows the subentry types that can be specified. These subentries are described in more detail in Chapter 35.

#### *subentry-list*

specifies the type of subentry list to be updated. "UCL Statement Syntax" shows the subentry types that can be specified. These subentries are described in more detail in Chapter 35.

#### *subentry-indicator*

specifies the subentry indicator to be updated. "UCL Statement Syntax" shows the subentry types that can be specified. These subentries are described in more detail in Chapter 35.

## UCL Statement Syntax

The UCL syntax descriptions in this chapter are arranged in alphabetical order. Table 23 on page 358 shows which entries can be processed in which zones and data sets.

*Table 23. SMP/E Entries That Can Be Processed by UCLIN*

<b>Entry Type</b>	<b>DLIB Zone</b>	<b>Target Zone</b>	<b>Global Zone</b>	<b>Other Data Set</b>
ASSEM	Yes	Yes		
BACKUP				SMPSCDS
Data element entries	Yes	Yes		
DDDEF	Yes	Yes	Yes	
DLIB	Yes	Yes		
DLIBZONE	Yes			
FMIDSET			Yes	
GLOBALZONE			Yes	
HFS	Yes	Yes		
LMOD	Yes	Yes		
MAC	Yes	Yes		
MOD	Yes	Yes		
MTSMAC				SMPMTS
OPTIONS			Yes	
SRC	Yes	Yes		
STSSRC				SMPSTS
SYSMOD	Yes	Yes	Yes	
TARGETZONE		Yes		
UTILITY			Yes	
ZONESET			Yes	

Not all the UCL statements can be used for each entry type. Table 24 shows which UCL statements can be used for entries in which SMP/E data sets.

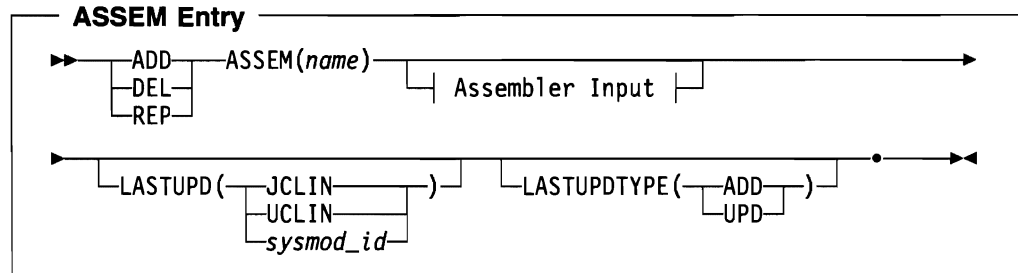
*Table 24. UCL Statements for SMP/E Data Sets*

<b>Data Set</b>	<b>ADD</b>	<b>DEL</b>	<b>REP</b>
SMPCSI	Yes	Yes	Yes
SMPMTS		Yes	
SMPSCDS		Yes	
SMPSTS		Yes	

This chapter shows only the syntax of UCL statements used to process entries. See Chapter 35 for additional information about each entry, such as:

- A description of the entry and its subentries
- LIST examples
- UNLOAD examples
- UCLIN examples

## ASSEM Entry Syntax (Distribution and Target Zone)



### Assembler Input:

```

|
|  ++ASMIN
|  ...
|  ... assembler input
|  ...
|  ++ENDASMIN
|

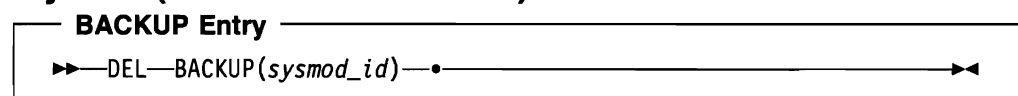
```

### Notes:

1. After UCLIN changes are done, the ASSEM entry must contain at least these subentries, unless the entire entry has been deleted:
  - ++ASMIN and ++ENDASMIN statements
  - The associated assembler input
2. The ++ASMIN and ++ENDASMIN statements must start in column 1.

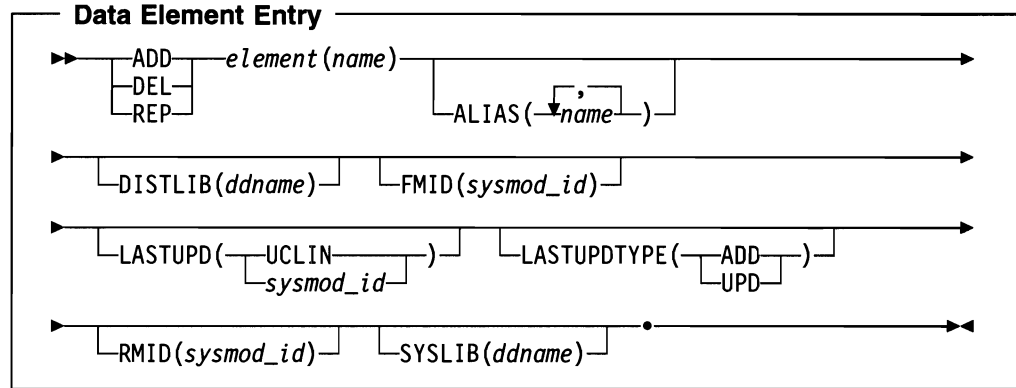
For a description of the subentries in the distribution or target zone ASSEM entry, see "ASSEM Entry (Distribution and Target Zone)" on page 640.

## BACKUP Entry Syntax (SMPSCDS Data Set)



For a description of the BACKUP entry, see "BACKUP Entries (SMPSCDS)" on page 645.

## Data Element Entry Syntax (Distribution and Target Zone)



### Notes:

1. After UCLIN changes are done, the data element entry must contain at least these subentries, unless the entire entry has been deleted:
  - DISTLIB
  - FMID
  - RMID
2. Table 26 on page 523 shows the types of data elements that can be specified for the *element* operand.
3. Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *element* operand might not contain any *xxx* value.) Table 27 on page 526 shows the *xxx* values and the languages they represent.

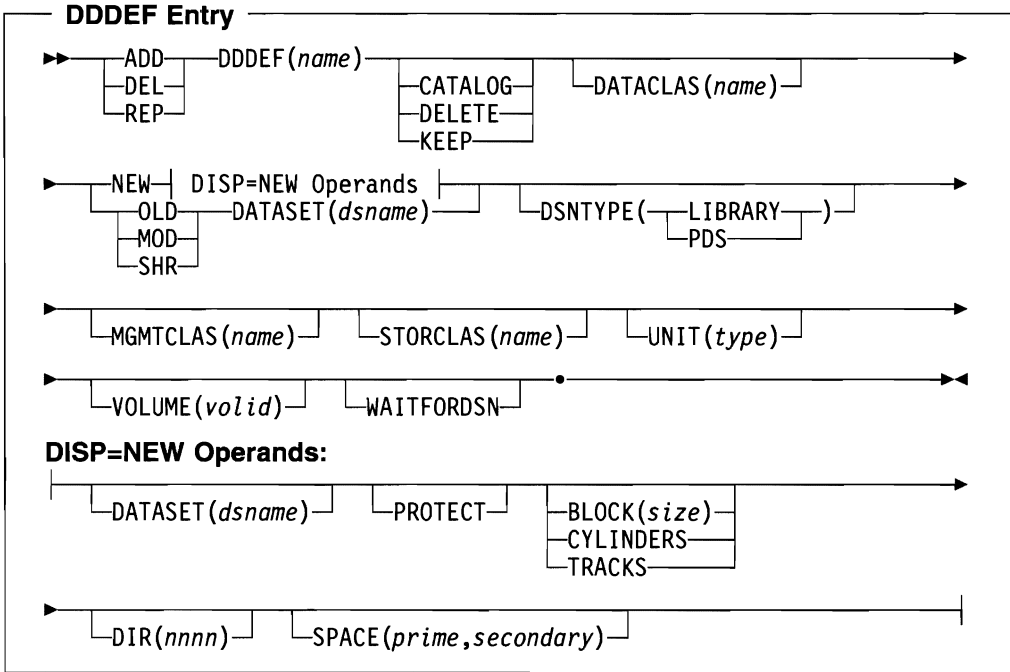
For a description of the subentries in data element entries, see “Data Element Entry (Distribution and Target Zone)” on page 648.

# DDDEF Entry Syntax (Distribution, Target, and Global Zone)

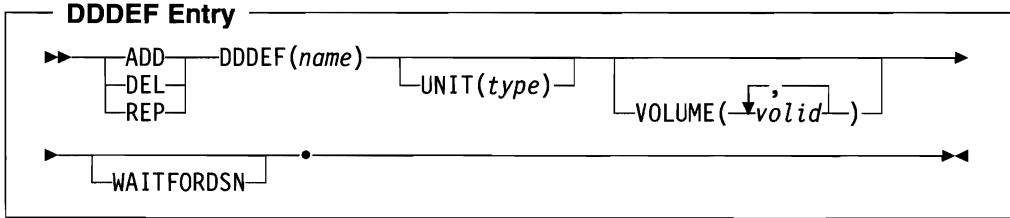
A separate syntax diagram is provided for each of the following types of data sets:

- Individual data set other than SMPTLIB or SYSOUT
- SMPTLIB data set in a distribution zone or target zone
- SMPTLIB data set in the global zone
- SYSOUT data set
- Concatenated data sets
- Path in a hierarchical file system (HFS)

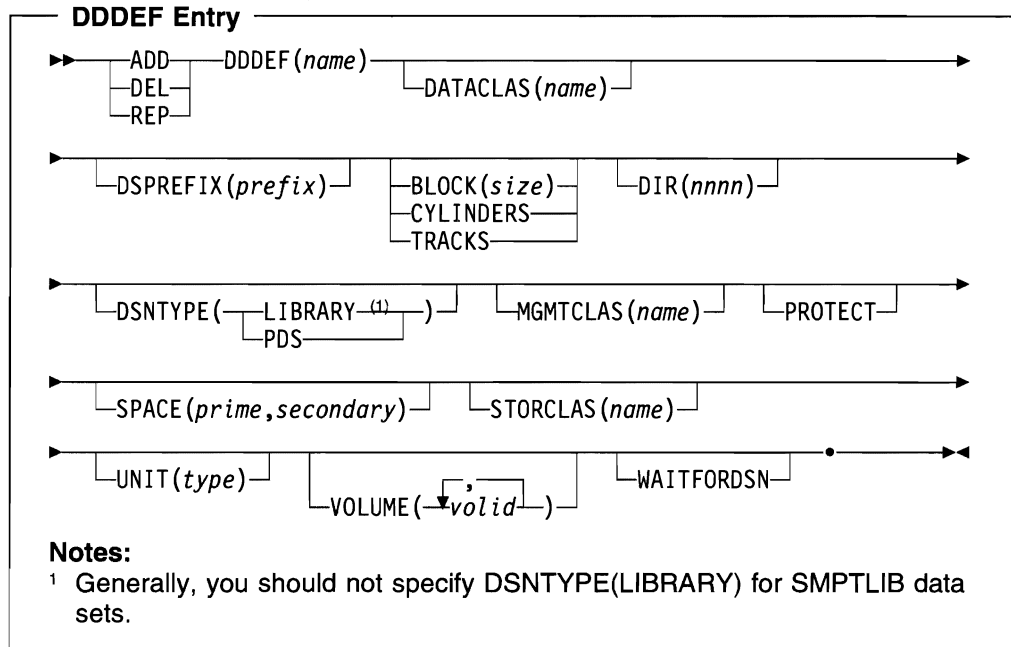
## Individual Data Set Other Than SMPTLIB or SYSOUT



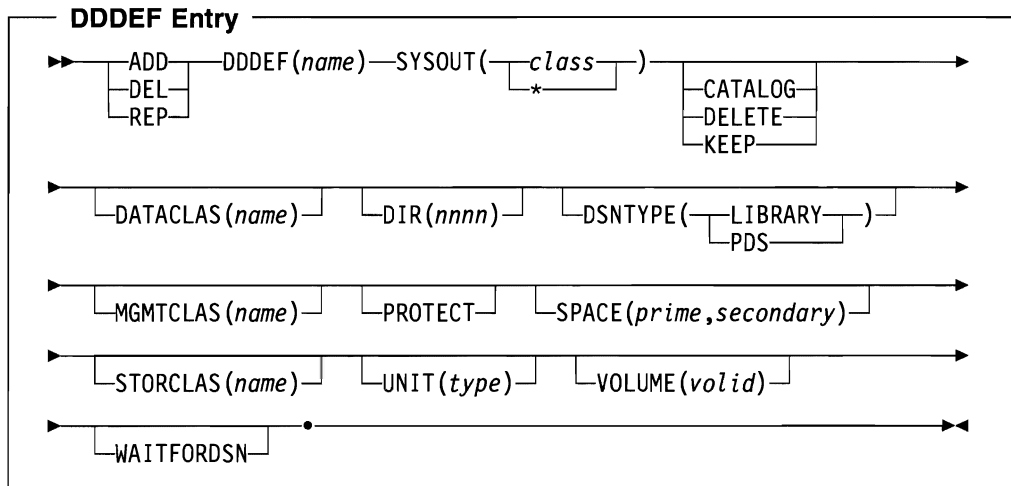
## SMPTLIB Data Set (Distribution or Target Zone)



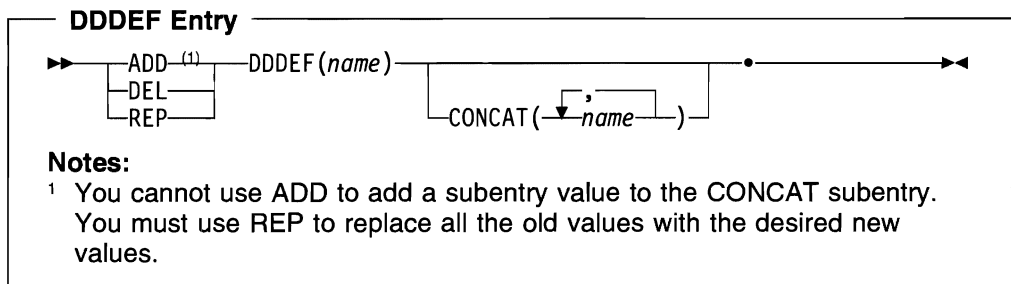
### SMPTLIB Data Set (Global Zone)



### SYSOUT Data Set

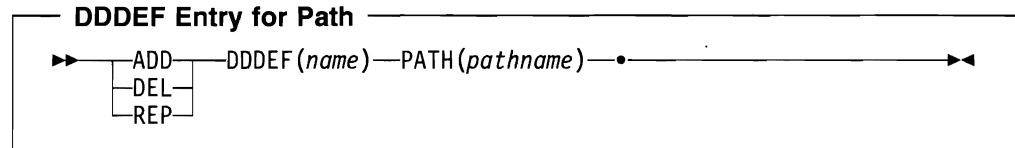


### Concatenated Data Sets



## Path in a Hierarchical File System (HFS)

Although a PATH subentry can be defined for a DDDEF entry in any type of zone, it is meaningful only in a target zone, because the information is used only when processing a target zone.

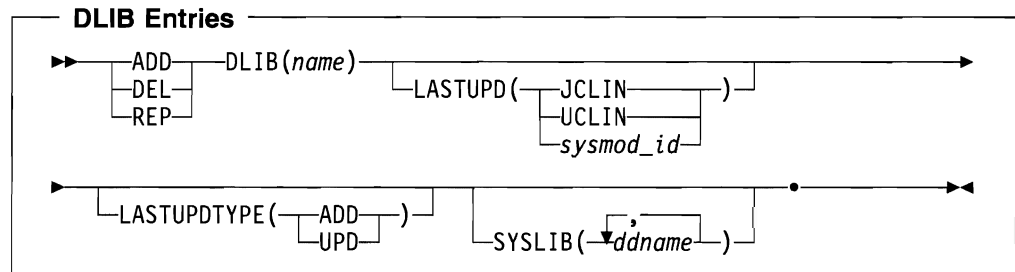


### Notes Except for Concatenated Data Sets and Paths:

1. **BLOCK** can also be specified as **BLK**.
2. **CYLINDERS** can also be specified as **CYL**.
3. **DATASET** can also be specified as **DA**.
4. Generally, you should not specify **DSNTYPE(LIBRARY)** for SMPTLIB data sets.
5. **DSPREFIX** may only be specified in a global zone DDDEF entry.
6. **TRACKS** can also be specified as **TRK**.
7. **WAITFORDSN** can also be specified as **WAIT**.

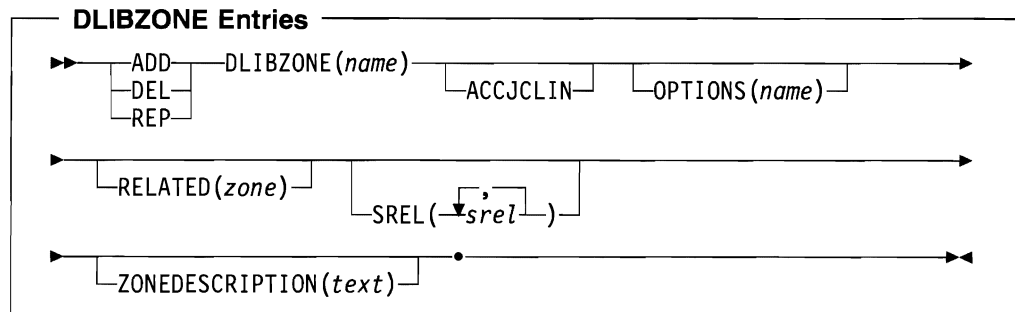
For a description of the subentries in DDDEF entries, see “DDDEF Entry (Distribution, Target, and Global Zone)” on page 654.

## DLIB Entry Syntax (Distribution and Target Zone)



For a description of the subentries in the distribution or target zone DLIB entry, see “DLIB Entry (Distribution and Target Zone)” on page 670.

## DLIBZONE Entry Syntax (Distribution Zone)

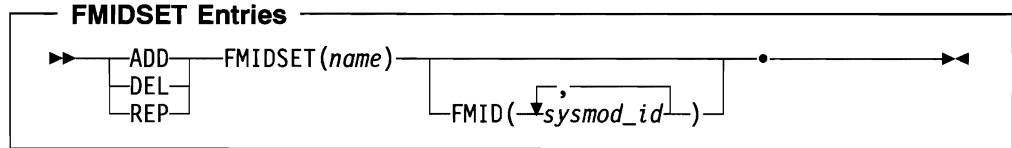


**Notes:**

1. **DLIBZONE** can also be specified as **DZONE**.
2. **ZONEDESCRIPTION** can also be specified as **ZDESC**.

For a description of the subentries in the distribution zone **DLIBZONE** entry, see “**DLIBZONE** Entry (Distribution Zone)” on page 675.

**FMIDSET Entry Syntax (Global Zone)**



**Notes:**

1. After UCLIN changes are made, the **FMIDSET** entry must contain at least the **FMID** subentries, unless the entire entry has been deleted.
2. **FMIDSET** can also be specified as **FMSET**.

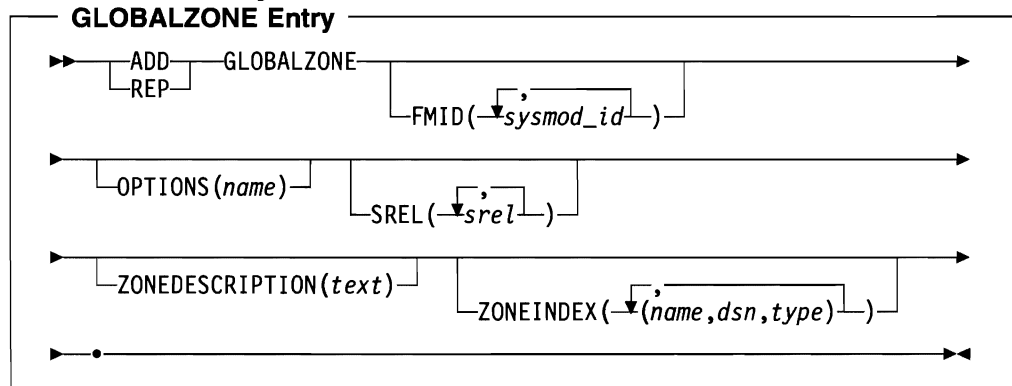
For a description of the subentries in the **FMIDSET** entry, see “**FMIDSET** Entry (Global Zone)” on page 679.

**GLOBALZONE Entry Syntax (Global Zone)**

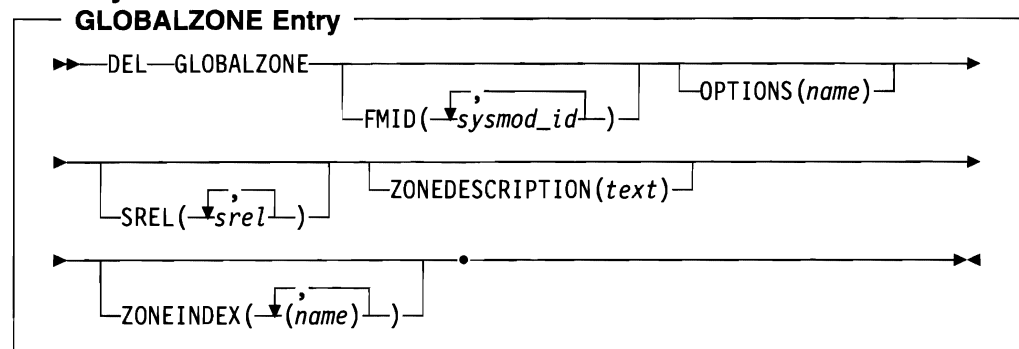
A separate syntax diagram is provided for:

- **ADD** and **REP** commands
- **DEL** commands

**ADD and REP Syntax**

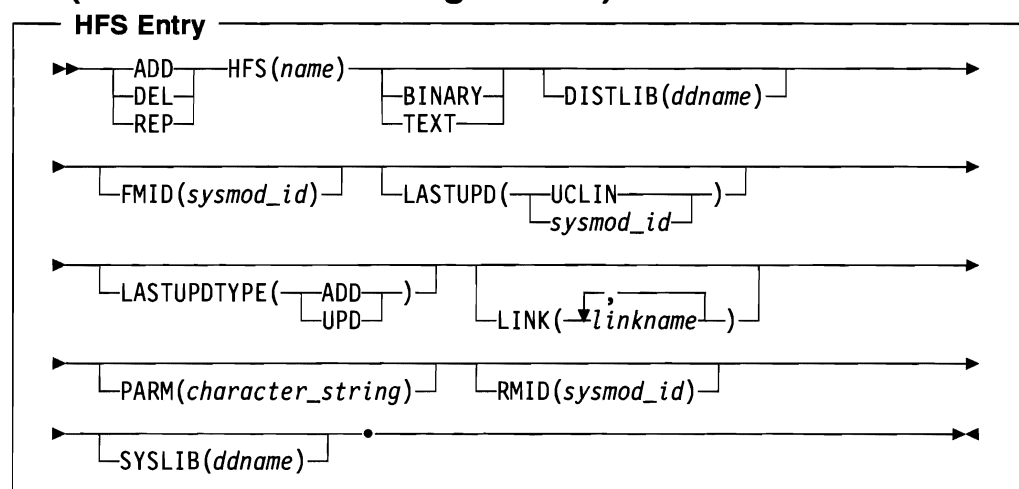




**DEL Syntax****Notes for ADD, DEL, and REP Syntax:**

1. After UCLIN changes are made, the GLOBALZONE entry must contain at least one of these subentries, unless the entire entry has been deleted:
  - FMID
  - OPTIONS
  - SREL
  - ZONEINDEX
2. GLOBALZONE can also be specified as GZONE.
3. ZONEDESCRIPTION can also be specified as ZDESC.
4. ZONEINDEX can also be specified as ZINDEX.

For a description of the subentries in the GLOBALZONE entry, see “GLOBALZONE Entry (Global Zone)” on page 682.

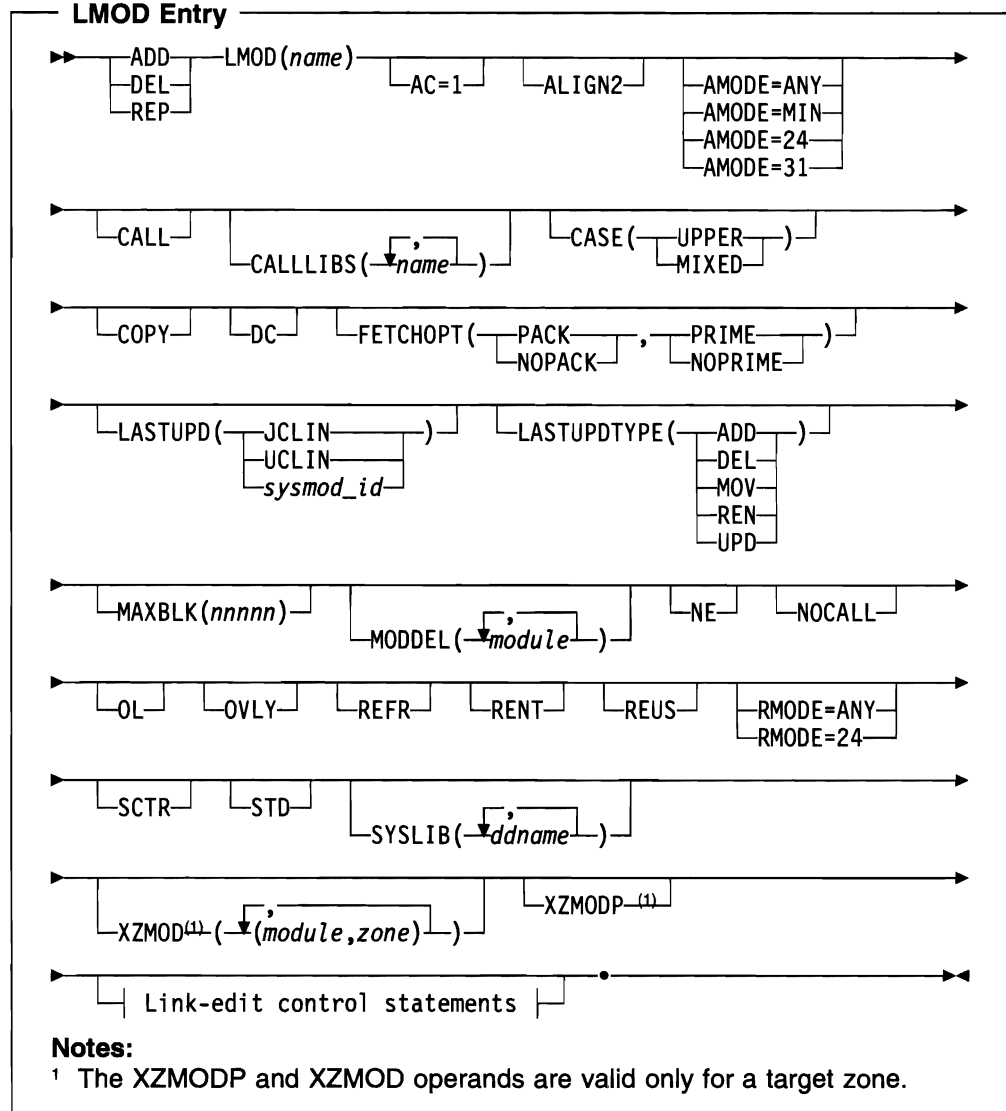
**HFS Entry Syntax (Distribution and Target Zone)**

**Note:** After UCLIN changes are made, the HFS entry must contain at least these subentries, unless the entire entry has been deleted:

- DISTLIB
- FMID
- RMID
- SYSLIB

For a description of the subentries in the HFS entry, see "HFS Entry (Distribution and Target Zone)" on page 687.

### LMOD Entry Syntax (Distribution and Target Zone)



**Link-Edit Control Statements:**

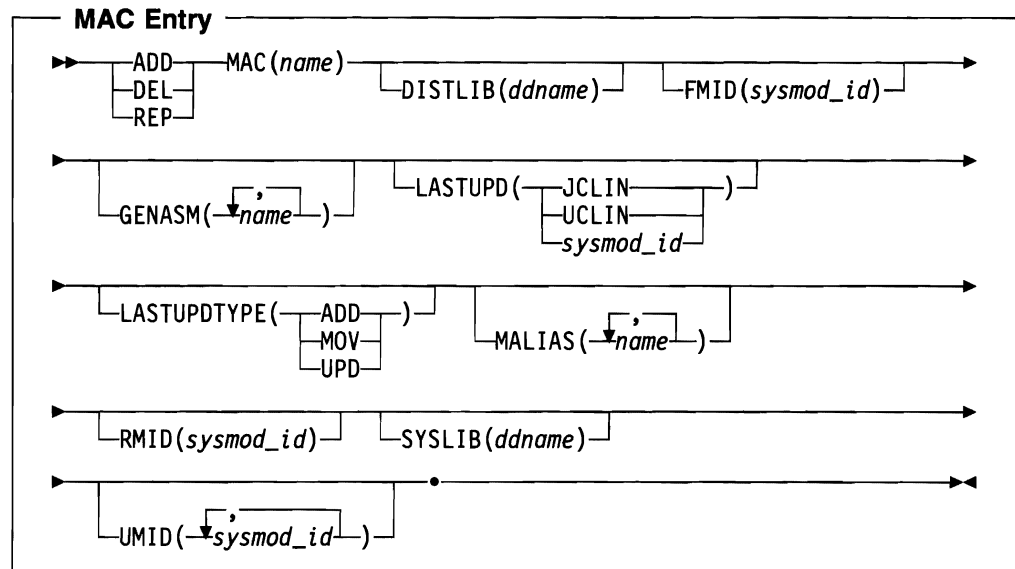
```

++LMODIN
...
... Link-edit control statements
...
++ENDLMODIN
  
```

**Notes:**

1. After UCLIN changes are made, the LMOD entry must contain at least the SYSLIB subentries, unless the entire entry has been deleted.
2. XZMOD and XZMODP subentries are valid only for target zone entries.
3. **AMODE=24** can also be specified as **AMOD=24**.
4. **AMODE=31** can also be specified as **AMOD=31**.
5. **AMODE=ANY** can also be specified as **AMOD=ANY**.
6. **AMODE=MIN** can also be specified as **AMOD=MIN**.
7. **NOCALL** can also be specified as **NCAL**.
8. **RMODE=24** can also be specified as **RMOD=24**.
9. **RMODE=ANY** can also be specified as **RMOD=ANY**.
10. The ++LMODIN and ++ENDLMODIN statements must start in column 1.
11. The link-edit control statements must start in or after column 2.

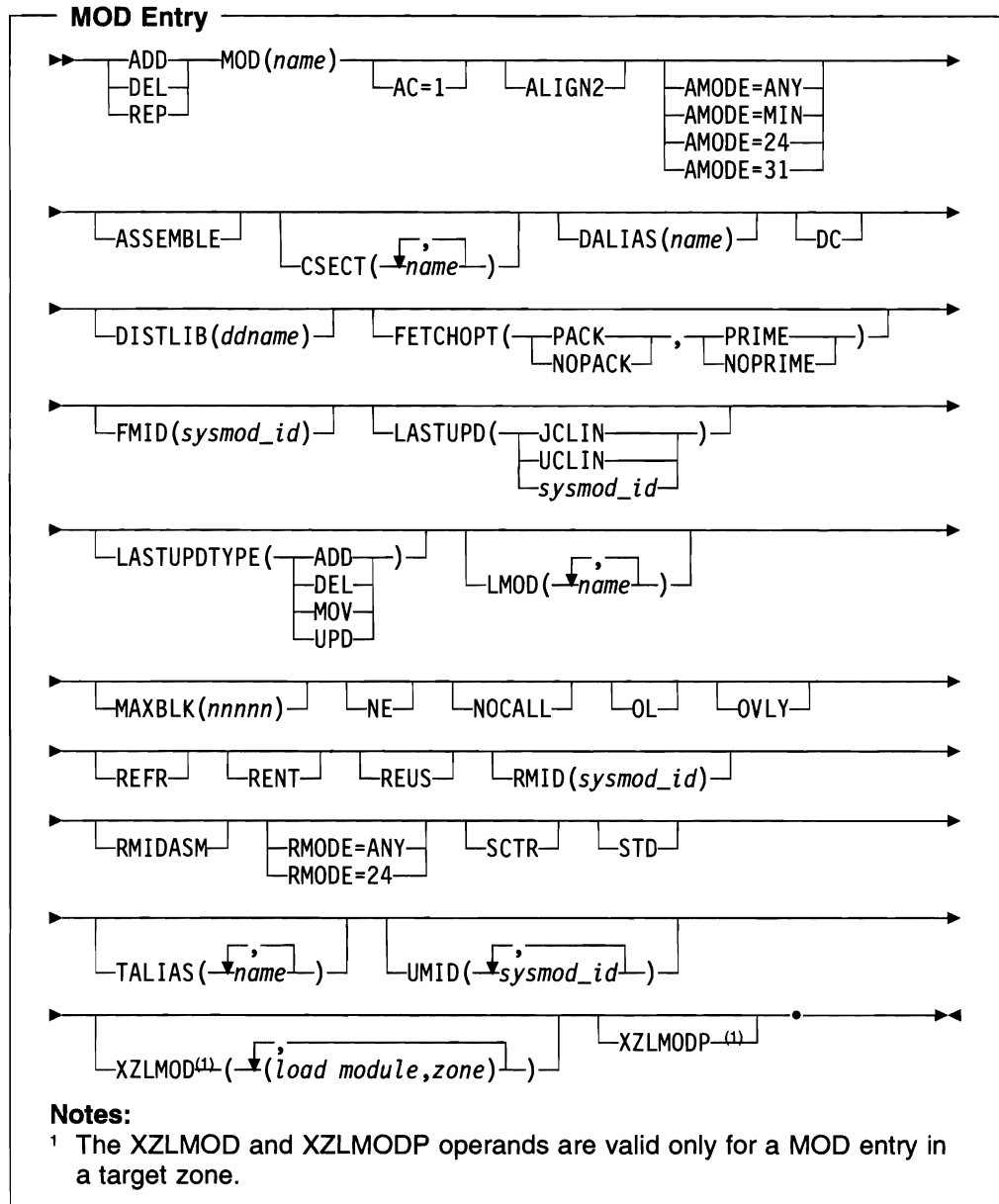
For a description of the subentries in the distribution or target zone LMOD entry, see “LMOD Entry (Distribution and Target Zone)” on page 698.

**MAC Entry Syntax (Distribution and Target Zone)****Notes:**

1. After UCLIN changes are made, the MAC entry must contain at least these subentries, unless the entire entry has been deleted:
  - FMID
  - RMID
2. **GENASM** can also be specified as **ASSEM**.

For a description of the subentries in the MAC entry, see “MAC Entry (Distribution and Target Zone)” on page 715.

## MOD Entry Syntax (Distribution and Target Zone)



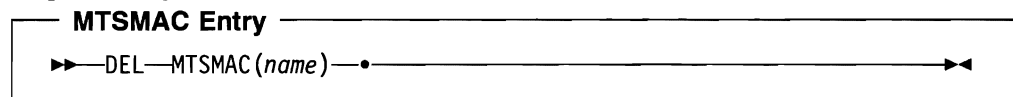
### Notes:

- After UCLIN changes are made, the MOD entry must contain at least these subentries, unless the entire entry has been deleted:
  - DISTLIB
  - FMID
  - RMID
- XZLMOD and XZLMODP subentries are valid only for target zone entries.
- ALIGN2 can also be specified as ALN2.
- AMODE=24 can also be specified as AMOD=24.
- AMODE=31 can also be specified as AMOD=31.

6. **AMODE=ANY** can also be specified as **AMOD=ANY**.
7. **AMODE=MIN** can also be specified as **AMOD=MIN**.
8. **NOCALL** can also be specified as **NCAL**.
9. **RMODE=24** can also be specified as **RMOD=24**.
10. **RMODE=ANY** can also be specified as **RMOD=ANY**.

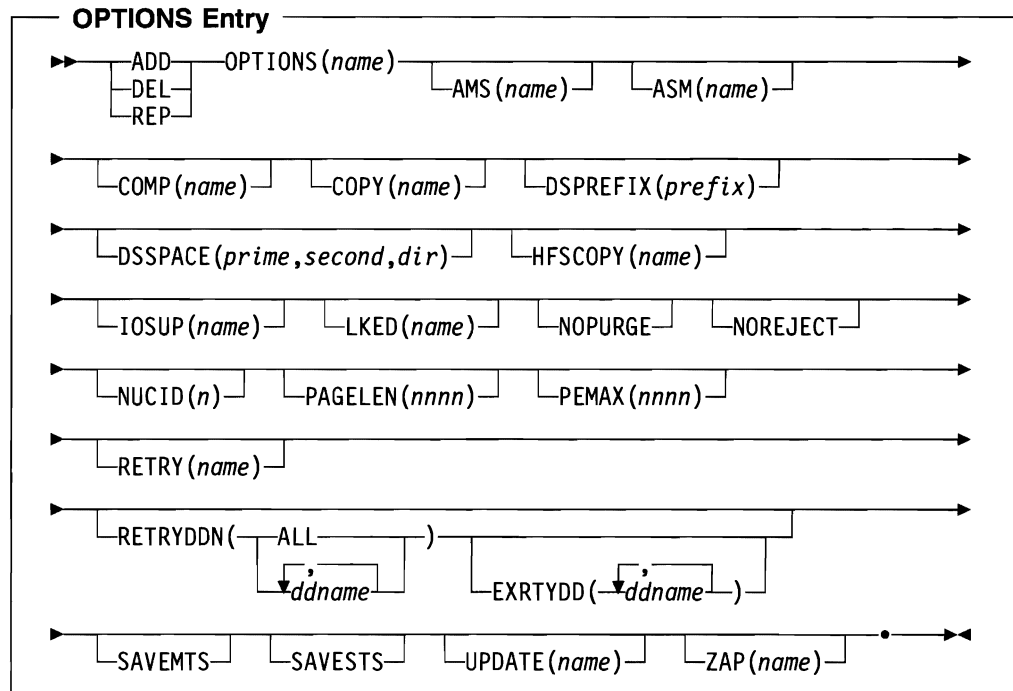
For a description of the subentries in the MOD entry, see “MOD Entry (Distribution and Target Zone)” on page 723.

## MTSMAC Entry Syntax (SMPMTS Data Set)



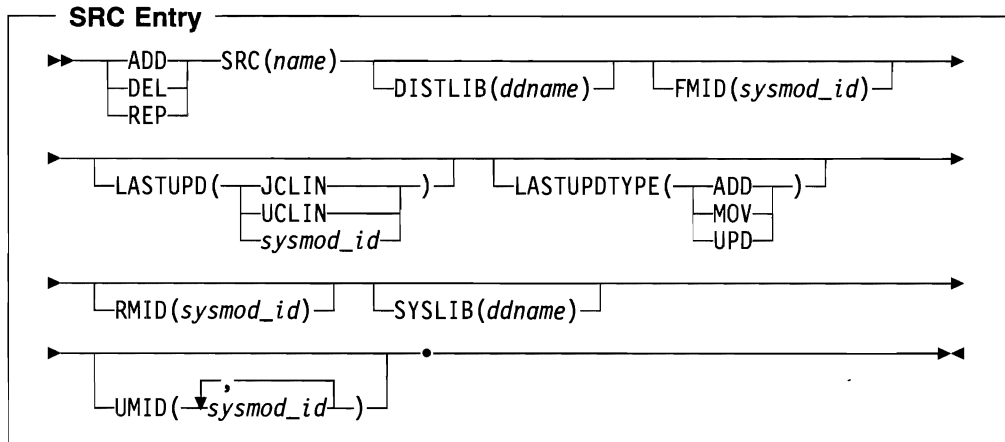
For a description of the MTSMAC entry, see “MTSMAC Entry (SMPMTS)” on page 739.

## OPTIONS Entry Syntax (Global Zone)



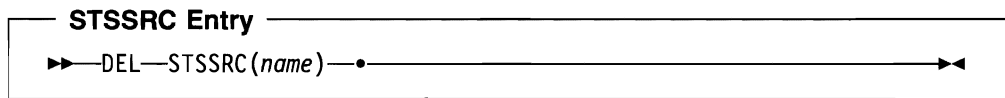
For a description of the subentries in the OPTIONS entry, see “OPTIONS Entry (Global Zone)” on page 741.

## SRC Entry Syntax (Distribution and Target Zone)



For a description of the subentries in the SRC entry, see “SRC Entry (Distribution and Target Zone)” on page 750.

## STSSRC Entry Syntax (SMPSTS Data Set)



For a description of the STSSRC entry, see “STSSRC Entry (SMPSTS)” on page 757.

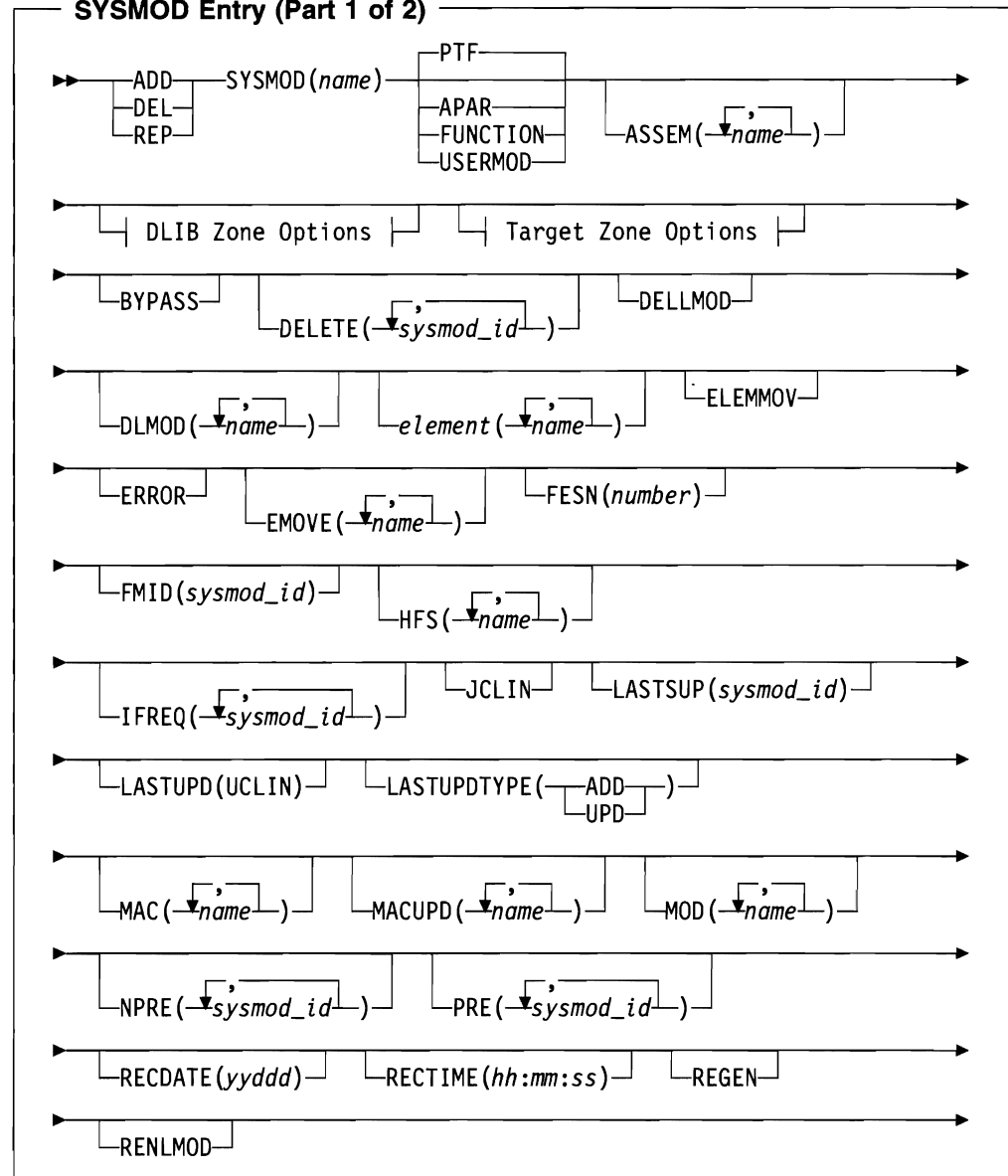
## SYSMOD Entry Syntax (Distribution and Target Zone)

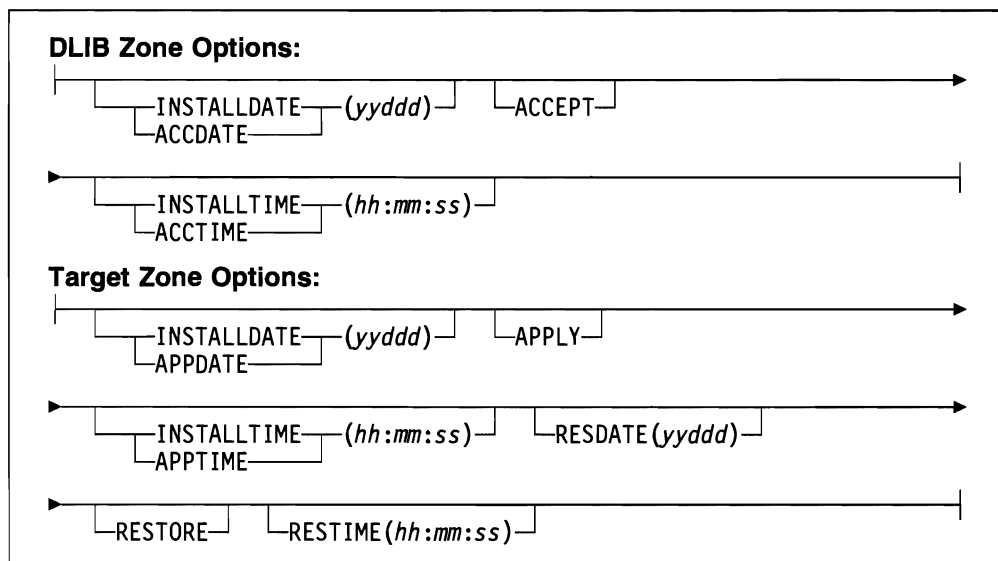
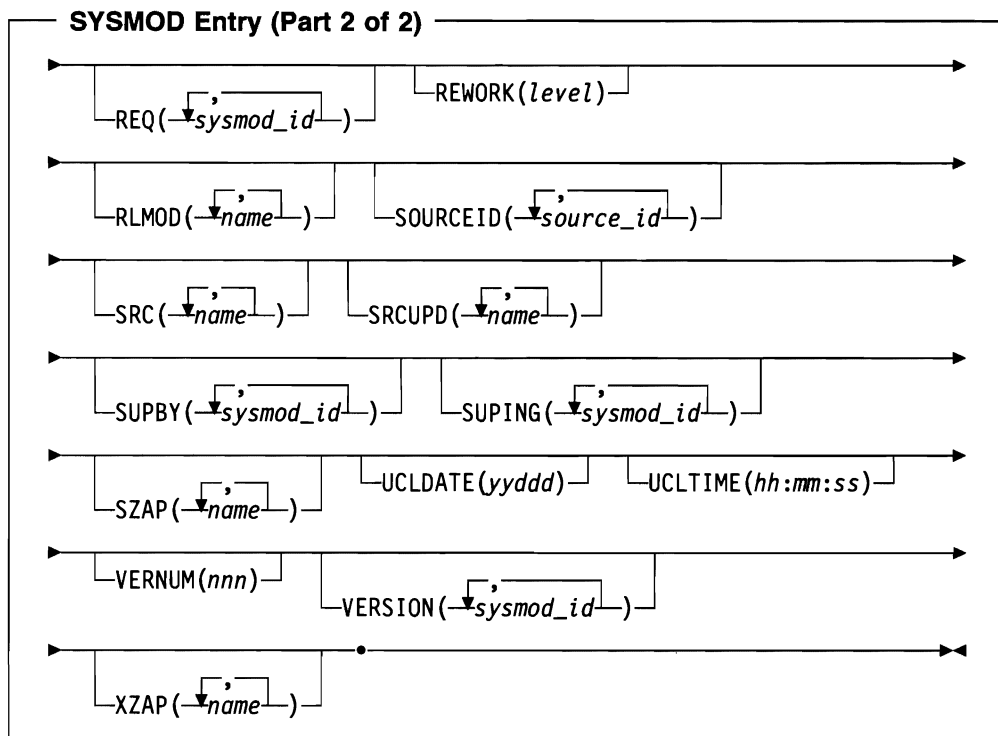
A separate syntax diagram is provided for each of the following types of SYSMOD entries:

- SYSMOD entry other than for a deleted SYSMOD
- SYSMOD entry containing the CIFREQ operand
- SYSMOD entry for deleted SYSMOD

## SYSMOD Entry Other Than Deleted SYSMOD

### SYSMOD Entry (Part 1 of 2)





**Notes:**

- Generally, after UCLIN changes are made, the SYSMOD entry must contain at least these subentries, unless the entire entry has been deleted:
  - ACCEPT or APPLY
  - APAR, FUNCTION, PTF, or USERMOD
  - INSTALLDATE
  - RECDATE
  - FMID

However, the entry for a deleted SYSMOD can contain only the DELBY sub-entry. The entry for a superseded SYSMOD can contain only the SUPBY sub-



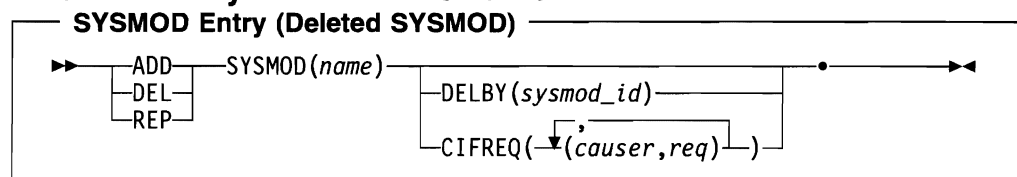
entry. The entry for a SYSMOD that is both deleted and superseded can contain only the DELBY and SUPBY subentries.

2. ACCDATE, ACCEPT, and ACCTIME can be used only in a distribution zone SYSMOD entry.
  - **ACCEPT** can also be specified as **ACPT** or **ACC**.
  - **INSTALLDATE** can be specified instead of **ACCDATE**. **INSTALLDATE** can also be specified as **INSDATE**.
  - **INSTALLTIME** can be specified instead of **ACCTIME**. **INSTALLTIME** can also be specified as **INSTIME**.
3. APPDATE, APPLY, and APPTIME can be used only in a target zone SYSMOD entry.
  - **APPLY** can also be specified as **APPL** or **APP**.
  - **INSTALLDATE** can be specified instead of **APPDATE**. **INSTALLDATE** can also be specified as **INSDATE**.
  - **INSTALLTIME** can be specified instead of **APPTIME**. **INSTALLTIME** can also be specified as **INSTIME**.
4. The *element* operand represents data elements. Table 26 on page 523 shows the types of data elements that can be specified for the *element* operand.
 

Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *element* operand does not contain any *xxx* value.) Table 27 on page 526 shows the *xxx* values and the languages they represent.
5. **ERROR** can also be specified as **ERR**.
6. **REGEN** can also be specified as **RGN**.
7. RESDATE, RESTORE, and RESTIME can be used only in a target zone SYSMOD entry.
8. **RESTORE** can also be specified as **REST** or **RES**.
9. **SUPBY** can also be specified as **SUP**.
10. The CIFREQ operand is mutually exclusive with all other UCL operands.

For a description of the subentries in the distribution or target zone SYSMOD entry, see “SYSMOD Entry (Distribution and Target Zone)” on page 759.

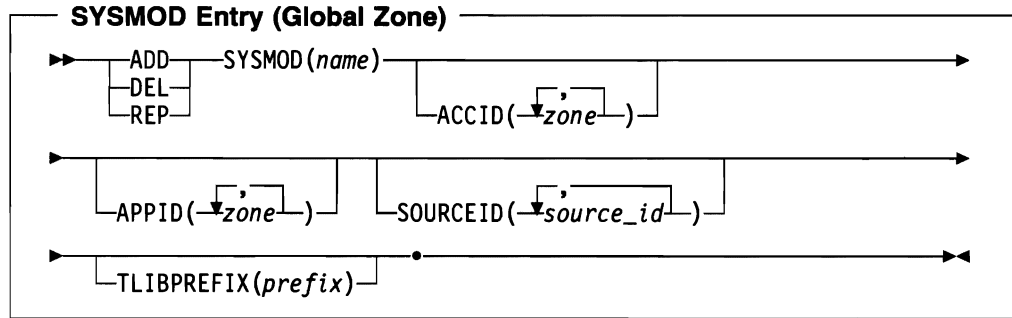
### SYSMOD Entry for Deleted SYSMOD



**Note:** The DELBY and CIFREQ operands are mutually exclusive.

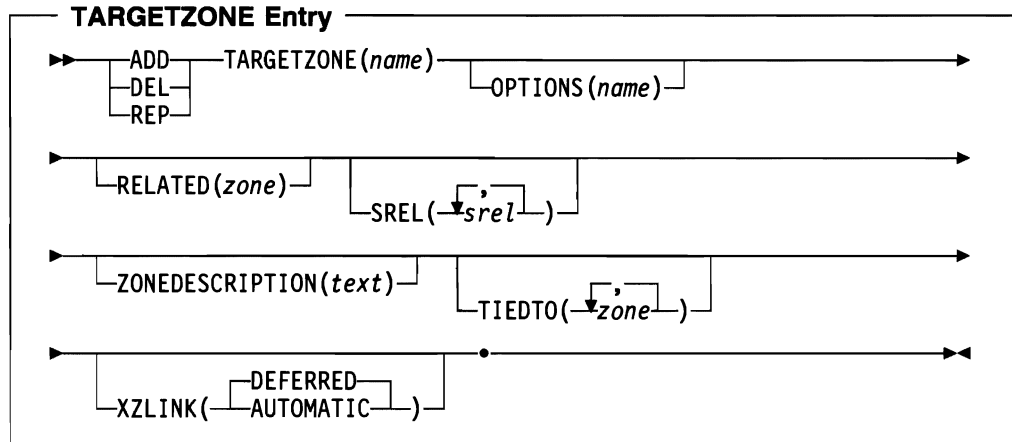
For a description of the subentries in the distribution or target zone SYSMOD entry, see “SYSMOD Entry (Distribution and Target Zone)” on page 759.

### SYSMOD Entry Syntax (Global Zone)



For a description of the subentries in the global zone SYSMOD entry, see “SYSMOD Entry (Global Zone)” on page 774. Note that only a limited subset of these subentries can be modified with UCLIN.

### TARGETZONE Entry Syntax (Target Zone)

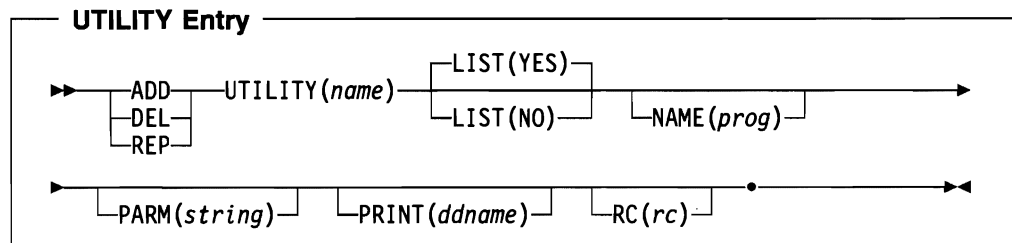


**Notes:**

1. `TARGETZONE` can also be specified as `TZONE`.
2. `ZONEDescription` can also be specified as `ZDESC`.

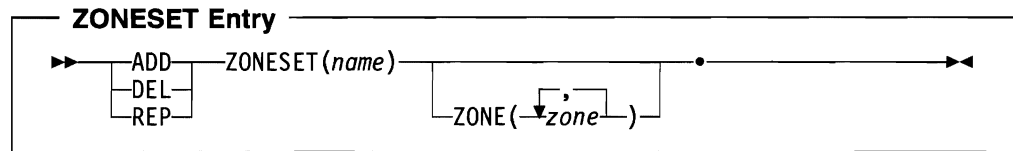
For a description of the subentries in the TARGETZONE entry, see “TARGETZONE Entry (Target Zone)” on page 782.

### UTILITY Entry Syntax (Global Zone)



For a description of the subentries in the UTILITY entry, see “UTILITY Entry (Global Zone)” on page 787.

## ZONESET Entry Syntax (Global Zone)



For a description of the subentries in the ZONESET entry, see “ZONESET Entry (Global Zone)” on page 794.

## Data Sets Used

The following data sets may be needed to run the UCLIN command. They can be defined by DD statements or, normally, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMPCSI	SMPLOGA	SMPPTS	SMPSCDS
SMPCNTL	SMPMTS	SMPSNAP	<i>zone</i>
SMPLOG	SMPOUT	SMPSTS	

### Notes:

1. SMPMTS is required if the MTSMAC entry is specified on a UCL statement.
2. SMPSTS is required if the STSSRC entry is specified on a UCL statement.
3. SMPSCDS is required if BACKUP entries are specified on a UCL statement.
4. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated through the ZONEINDEX information in the GLOBALZONE entry.

## Output

The File Allocation report is produced during UCLIN processing. It is described in Chapter 31.

## Usage Notes

- If you want to add a subentry or a subentry value, you should use the ADD statement. Although, in some cases, you may get the desired results using the REP statement, in other cases, the results may not be what you expected. For more information, see “Processing” on page 377.
- If you want to delete a subentry or a subentry value, do not try to do it by adding a null value. For example, do not use `ADD subentry()`. Use the DEL statement instead.
- To delete a subentry, you must know and specify an existing value. However, for subentry lists, if you want to delete all the values but do not know what they all are, you can specify the subentry list operand followed by a left and right

parenthesis. For example, to delete all MOD values in the SYSMOD entry for UR11111, you could use these commands:

```

SET      BDY(TGT1)      /* Set to target zone.      */.
UCLIN                                /*                          */.
DEL      SYSMOD(UR11111)/* Delete data from SYSMOD: */
          MOD()          /* all MOD subentries.     */.
          /*                          */.
ENDUCL                                /*                          */.
    
```

The parentheses, “()”, mean that SMP/E should delete the subentry list they enclose, even though no value is specified.

**Note:** This technique is intended only for subentry lists. Do not use it for single-value subentries.

## Examples

The following general examples are provided to help you use the UCLIN command:

- “Example 1: UCLIN to Change a Global Zone Entry”
- “Example 2: UCLIN to Change a Target Zone Entry”
- “Example 3: UCLIN to Change a Distribution Zone Entry”

For specific examples of UCLIN for each entry type, see Chapter 35.

### Example 1: UCLIN to Change a Global Zone Entry

You can use the following commands to change global zone entries:

```

SET      BDY(GLOBAL)    /* Set to global zone.      */.
UCLIN                                /*                          */.
...                                           /*                          */.
... UCL statements for global zone entries
...                                           /*                          */.
ENDUCL                                /*                          */.
    
```

### Example 2: UCLIN to Change a Target Zone Entry

You can use the following commands to change target zone entries:

```

SET      BDY(TGT1)      /* Set to target zone.      */.
UCLIN                                /*                          */.
...                                           /*                          */.
... UCL statements for target zone entries
...                                           /*                          */.
ENDUCL                                /*                          */.
    
```

### Example 3: UCLIN to Change a Distribution Zone Entry

You can use the following commands to change distribution zone entries:

```

SET      BDY(DLIB1)     /* Set to DLIB zone.       */.
UCLIN                                /*                          */.
...                                           /*                          */.
... UCL statements for DLIB zone entries
...                                           /*                          */.
ENDUCL                                /*                          */.
    
```

## Processing

SMP/E processes each UCL statement and each operand on each UCL statement as they are encountered. Table 25 shows how these statements are processed.

Table 25. How SMP/E Processes UCL Statements

Type Specified	ADD	DEL	REP
Entry	If the entry does not exist, SMP/E creates a new one.  If the entry exists, SMP/E assumes a new subentry or subentry value is being added.	If the entry exists, SMP/E deletes it.  Otherwise, an error occurs.	If the entry does not exist, SMP/E creates a new one.  If the entry exists, SMP/E assumes that a new subentry or subentry value is being replaced.
Subentry	If the subentry does not exist in the entry, SMP/E adds the new subentry.  Otherwise, an error occurs.	If the subentry exists in the entry, SMP/E deletes the subentry.  Otherwise, an error occurs.	If the subentry exists in the entry, SMP/E replaces the current value with the new value.  If the subentry does not exist in the entry, SMP/E adds the new subentry.
Subentry list or subentry list value	If the subentry list does not exist in the entry, SMP/E adds it.  Likewise, if the subentry list value does not exist in the entry, SMP/E adds it.  Otherwise, an error occurs.	If the subentry list value exists in the entry, SMP/E deletes that value.  If all the existing subentry list values were specified, or if a null value was specified, SMP/E deletes the entire subentry.  Otherwise, an error occurs.	SMP/E does not replace individual values in a subentry list.  If the subentry list exists in the entry, SMP/E replaces all the current values with the new value.  If the subentry list does not exist in the entry, SMP/E adds it.
<b>Notes for subentry list or subentry list value:</b>			
<ul style="list-style-type: none"> <li>• ADD cannot be used to add a subentry value to a CONCAT subentry list for a DDDEF entry, because the order of values in the subentry list is important for CONCAT. Adding a single value to an existing list does not provide enough information as to how the whole list should be ordered.</li> <li>• REP must be used to logically add a new value to a CONCAT subentry list for a DDDEF entry, because the order of the subentry list values is important for CONCAT, and REP is the only provided method for specifying the desired order. (REP replaces the entire subentry list; therefore, you must specify all the existing subentry values, as well as the new subentry value.)</li> </ul>			
Subentry indicator	If the subentry indicator is not set, SMP/E sets it to "on."  Otherwise, an error occurs.	If the subentry indicator is set to "on," SMP/E resets it to "off."  Otherwise, an error occurs.	SMP/E sets the indicator to "on," regardless of its current setting.

For element and SYSMOD entries, SMP/E first copies the entry into storage. Then, as SMP/E encounters each operand, it updates the copy of the entry. When SMP/E reaches the end of the UCL statement, it makes sure the updated entry contains at least the minimum data required and that the data in that entry is consistent. If processing was successful, or if only informational or warning messages were issued, SMP/E replaces the original entry with the updated copy.

If an error occurs while one of the UCL statements is being processed, SMP/E does not make the change for that one statement. However, it continues to process subsequent UCL statements and, if they are valid, makes the requested changes. For each error, SMP/E issues an error message indicating the cause of the problem and deletes the copy of the entry. The original entry remains unchanged.

**Note:** For entries other than element entries or SYSMOD entries, if UCLIN deletes all the subentries for that entry, SMP/E deletes the entire entry.

For each element entry successfully changed by a UCL statement, SMP/E sets the LASTUPD subentry to **UCLIN** and the LASTUPDTYPE subentry to either **ADD** or **UPD**.

The return code for all the UCLIN processing is the highest return code from the processing of any of the UCL statement in the UCLIN input.

UCLIN processing stops when the ENDUCL command is encountered.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of UCLIN processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

**Note:** Either the global zone, target zone, or the distribution zone is accessed, based on the zone specified in the previous SET command.

### 2. UCLIN processing

Global zone	—	Update with exclusive enqueue.
SMPPTS	—	Update with exclusive enqueue.
Target zone	—	Update with exclusive enqueue.
DLIB zone	—	Update with exclusive enqueue.

**Note:** Either the global zone and SMPPTS, the target zone, or the distribution is accessed, based on the zone specified in the previous SET command.

### 3. Termination

All resources are freed.

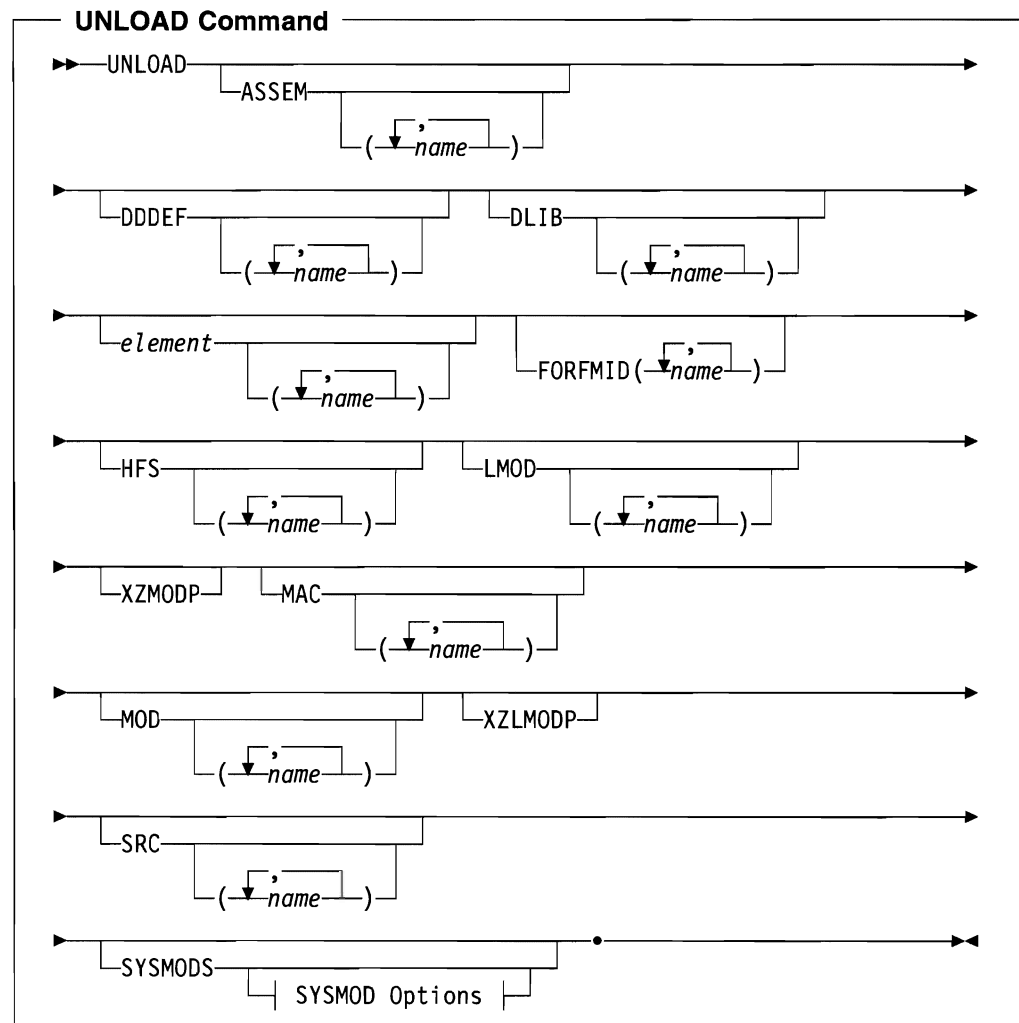
## Chapter 23. The UNLOAD Command

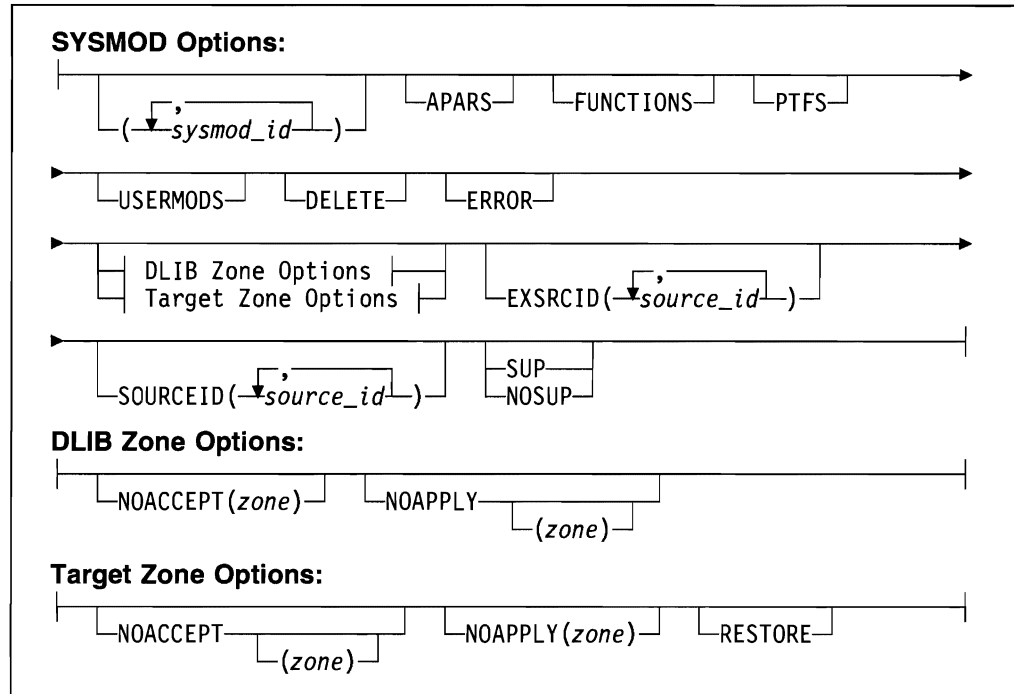
The UNLOAD command causes SMP/E to unload entries from the target zone or distribution zone to the SMPPUNCH data set. The output produced is in the form of UCL statements which, when processed by SMP/E, recreate the unloaded entries in the distribution zone or the target zone. This function allows the user to unload selected parts of a target zone or distribution zone for initialization of entries on other zones. Do **not** use UNLOAD to back up a zone. Use ZONECOPY or ZONEEXPORT/ZONEIMPORT instead.

### Zones for SET BOUNDARY

For the UNLOAD command, the SET BOUNDARY command must specify the name of the target or distribution zone containing the entries to be unloaded. You must ensure that the data you request to be unloaded is valid for the zone type specified.

### Syntax





### Notes:

1. The SYSMODS operand is optional if you specify any of the following operands:

APARS	NOACCEPT	RESTORE
DELETE	NOAPPLY	SOURCEID
ERROR	NOSUP	SUP
EXSRCID	PTFS	USERMODS
FUNCTIONS		

2. The XZLMODP and XZMODP operands are valid only for target zone entries.

See the operand descriptions for details.

## Operands

### APARS

indicates that SMP/E should unload APAR SYSMODs.

#### Notes:

1. **APARS** can also be specified as **APAR**.
2. When **APARS** is used with **FUNCTIONS**, **PTFS**, or **USERMODS**, SMP/E unloads any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.

### ASSEM

indicates that SMP/E should unload all ASSEM entries or the specified ASSEM entries.



**DDDEF**

indicates that SMP/E should unload all DDDEF entries or the specified DDDEF entries.

**DELETE**

indicates that SMP/E should unload entries for function SYSMODs that have been explicitly deleted from the target zone or distribution zone by other function SYSMODs.

**Notes:**

1. **DELETE** can also be specified as **DEL**.
2. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though **SYSMOD** was also specified, even if it was not.

**DLIB**

indicates that SMP/E should unload all DLIB entries or the specified DLIB entries.

*element*

is used to unload a particular type of data element entry. It indicates that SMP/E should unload all data element entries of that type or the specified data element entries.

**Notes:**

1. Data element entries exist only in the target and distribution zones.
2. Table 26 on page 523 shows the types of data elements that can be specified for the *element* operand.
3. Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *element* operand does not contain any *xxx* value.) Table 27 on page 526 shows the *xxx* values and the languages they represent.

**ERROR**

indicates that SMP/E should unload SYSMOD entries in which the ERROR indicator is set.

**Notes:**

1. **ERROR** can also be specified as **ERR**.
2. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though **SYSMOD** was also specified, even if it was not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

**EXSRCID**

indicates that SYSMODs associated with the specified source IDs should **not** be unloaded.

### Notes:

1. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.
2. There are two ways to specify source IDs:
  - Explicitly, by specifying a particular source ID (for example, **PUT9303**). In this case, only that particular source ID is used.
  - Implicitly, by specifying either **\*** or **c\*** (for example, **PUT\***), where *c* is a 1- to 7-character string. In the second case, all source IDs that start with the specified character string are used.
3. A given source ID may be explicitly specified **only once** on the EXSRCID operand.
4. The same source ID may **not** be explicitly specified on both the EXSRCID and SOURCEID operands.
5. If a source ID is specified implicitly on the EXSRCID operand and also, either implicitly or explicitly, on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.
6. If a given SYSMOD has multiple source IDs, if at least one of those source IDs is specified either implicitly or explicitly on the SOURCEID operand, and if another one of its source IDs is specified either implicitly or explicitly on the EXSRCID operand, the SYSMOD is excluded from processing.  
  
For example, assume PTF UZ12345 has been assigned source IDs SMCREC and PUT9303. If you specify **SOURCEID(SMC\*)** and **EXSRCID(PUT9303)**, the SYSMOD is excluded from processing.
7. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

### FORFMID

indicates that SMP/E should unload only entries currently owned by one of the specified FMIDs or by an FMID defined in one of the specified FMIDSET entries.

### Notes:

1. You can specify FMIDs, FMIDSET entries, or both.
2. Only element and SYSMOD entries are unloaded by the FORFMID operand.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not, **unless** an element type operand was also specified. In that case, FORFMID limits the element entries that are unloaded.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

**FUNCTIONS**

indicates that SMP/E should unload function SYSMODs.

**Notes:**

1. **FUNCTIONS** can also be specified as **FUNCTION**.
2. When **FUNCTIONS** is used with **APARS**, **PTFS**, or **USERMODS**, SMP/E unloads any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.

**HFS**

indicates that SMP/E should unload all HFS entries or the specified HFS entries.

**LMOD**

indicates that SMP/E should unload all LMOD entries or the specified LMOD entries.

**MAC**

indicates that SMP/E should unload all MAC entries or the specified MAC entries.

**MOD**

indicates that SMP/E should unload all MOD entries or the specified MOD entries.

**NOACCEPT**

indicates that SMP/E should unload SYSMOD entries from the current zone that are not accepted into the specified distribution zone.

**Notes:**

1. **NOACCEPT** can also be specified as **NOACC**.
2. If a target zone is specified on the SET command and no distribution zone is specified on NOACCEPT, SMP/E uses the distribution zone from the RELATED subentry in the TARGETZONE entry.
3. If a distribution zone is specified on the SET command and no distribution zone is specified on NOACCEPT, SMP/E issues an error message.
4. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.
5. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

**NOAPPLY**

indicates that SMP/E should unload SYSMOD entries from the current zone that are not applied to the specified target zone.

### Notes:

1. **NOAPPLY** can also be specified as **NOAPP**.
2. If a distribution zone is specified on the **SET** command and no target zone is specified on **NOAPPLY**, SMP/E uses the target zone from the **RELATED** subentry in the **DLIBZONE** entry.
3. If a target zone is specified on the **SET** command and no target zone is specified on **NOAPPLY**, SMP/E issues an error message.
4. Because this operand describes the type of **SYSMOD** to be unloaded, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.
5. If no **SYSMOD** types are specified, all eligible **SYSMODs** are included. To process specific types of **SYSMODs**, you must specify the desired **SYSMOD** types.

### NOSUP

indicates that SMP/E should unload entries for **SYSMODs** that have not been superseded.

### Notes:

1. **NOSUP** is mutually exclusive with **SUP**.
2. Because this operand describes the type of **SYSMOD** to be unloaded, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.
3. If no **SYSMOD** types are specified, all eligible **SYSMODs** are included. To process specific types of **SYSMODs**, you must specify the desired **SYSMOD** types.

### PTFS

indicates that SMP/E should unload **PTF SYSMODs**.

### Notes:

1. **PTFS** can also be specified as **PTF**.
2. When **PTFS** is used with **APARS**, **FUNCTIONS**, or **USERMODS**, SMP/E unloads any **SYSMOD** whose type matches **any one** of those specified.
3. Because this operand describes the type of **SYSMOD** to be unloaded, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.

### RESTORE

indicates that SMP/E should unload **SYSMOD** entries in which the **RESTORE** indicator is set. These **SYSMODs** have been incompletely restored and are "in error."

### Notes:

1. **RESTORE** is allowed when the **SET** command specifies a target zone.
2. **RESTORE** can also be specified as **RES**.
3. Because this operand describes the type of **SYSMOD** to be unloaded, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.

4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

**SOURCEID**

indicates that SMP/E should unload only those SYSMOD entries associated with one of the specified SOURCEID values.

**Notes:**

1. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.
2. There are two ways to specify source IDs:
  - Explicitly, by specifying a particular source ID (for example, **PUT9303**). In this case, only that particular source ID is used.
  - Implicitly, by specifying either **\*** or **c\*** (for example, **PUT\***), where **c** is a 1- to 7-character string. In the second case, all source IDs that start with the specified character string are used.
3. A given source ID may be explicitly specified **only once** on the SOURCEID operand.
4. The same source ID may **not** be explicitly specified on both the EXSRCID and SOURCEID operands.
5. If a source ID is specified implicitly on the SOURCEID operand and also, either implicitly or explicitly, on the EXSRCID operand, all SYSMODs with that source ID are excluded from processing.
6. If a given SYSMOD has multiple source IDs, if at least one of those source IDs is specified either implicitly or explicitly on the SOURCEID operand, and if another one of its source IDs is specified either implicitly or explicitly on the EXSRCID operand, the SYSMOD is excluded from processing.  
  
For example, suppose PTF UZ12345 has been assigned source IDs SMCREC and PUT9303. If you specify **SOURCEID(SMC\*)** and **EXSRCID(PUT9303)**, the SYSMOD is excluded from processing.
7. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

**SRC**

indicates that SMP/E should unload all SRC entries or the specified SRC entries.

**SUP**

indicates that SMP/E should unload entries for SYSMODs that have been superseded.

### Notes:

1. SUP is mutually exclusive with NOSUP.
2. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though **SYSMOD** had also been specified, even if it has not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

### SYSMODS

indicates that SMP/E should unload all SYSMOD entries or the specified SYSMOD entries.

You can limit which SYSMOD entries are unloaded by coding one or more of the following SYSMOD qualifier operands:

APARS, FUNCTIONS, PTFS, or USERMODS  
DELETE  
ERROR  
EXSRCID  
FORFMID  
NOACCEPT  
NOAPPLY  
NOSUP or SUP  
RESTORE  
SOURCEID

### Notes:

1. **SYSMODS** can also be specified as **SYSMOD**.
2. If you specify any of the SYSMOD qualifier operands, SMP/E assumes that you want the SYSMOD entries unloaded and thus processes as if you had also entered SYSMOD.

### USERMODS

indicates that SMP/E should unload USERMODS.

### Notes:

1. **USERMODS** can also be specified as **USERMOD**.
2. When **USERMODS** is used with **APARS**, **FUNCTIONS**, or **PTFS**, SMP/E unloads any SYSMOD whose type matches **any one** of those specified.

### XZLMODP

indicates that SMP/E should unload MOD entries for all modules that have been linked into load modules controlled by a different target zone. (The MOD entries for these modules contain XZLMODP subentries.)

**Notes:**

1. XZLMODP is allowed only when the SET command specifies a target zone.
2. The appropriate MOD entries are unloaded, regardless of whether the MOD operand was specified on the UNLOAD command.
3. If both MOD and XZLMODP are specified, only MODs with cross-zone sub-entries are unloaded. If a list of MODs and XZLMODP are specified, all the specified MODs, as well as all the MODs with cross-zone subentries, are unloaded.

**XZMODP**

indicates that SMP/E should unload LMOD entries for all load modules containing modules from a different target zone. (The LMOD entries for these load modules contain XZMODP subentries.)

**Notes:**

1. XZMODP is allowed only when the SET command specifies a target zone.
2. The appropriate LMOD entries are unloaded, regardless of whether the LMOD operand was specified on the UNLOAD command.
3. If you specify both LMOD and XZMODP, only LMODs with cross-zone sub-entries are unloaded. If you specify a list of LMODs and XZMODP, all the specified LMODs, as well as all the LMODs with cross-zone subentries, are unloaded.

For examples of unloading each specific entry type, see the section for that entry in Chapter 35.

**Syntax Notes**

- Except where noted, you can specify multiple operands on a single UNLOAD command. This improves the overall performance of SMP/E.
- The order in which the entries are unloaded depends on their sequence in the appropriate SMP/E data set, not on the order specified on the UNLOAD command.
- You can mix mass requests and selective requests on the same UNLOAD command. For example:

```

SET      BDY(TGT1)      /* Set to target zone.      */
UNLOAD  MOD             /* Unload all modules,     */
        MAC(MAC01      /* only two macros,       */
          MAC02)      /*                          */
        SRC(SRC01      /* only two source,      */
          SRC02)      /*                          */
        DLIB           /* all DLIBs,            */
        DDDEF          /* all DDDEFs,           */
        SYSMOD(UZ00001 /* only these five SYSMODS. */
          UZ00002 /*                          */
          UZ00003 /*                          */
          UZ00004 /*                          */
          UZ00005)/*                          */

```

- If you specify a given SYSMOD on the SYSMODS operand, the SYSMOD is unloaded, regardless of whether it may be included or excluded by other operands.

---

### Data Sets Used

The following data sets may be needed to run the UNLOAD command. You can define them by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPPUNCH	SMPSNAP
SMP_CSI	SMP_OUT	SMPRPT	<i>zone</i>
SMPLOG			

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

---

### Output

UNLOAD output is written to the SMPPUNCH data set. For an example of the UNLOAD output for a particular entry type, see the section for that entry type in Chapter 35.

The following reports are produced during UNLOAD processing:

- File Allocation report
- UNLOAD Summary report

These reports are described in Chapter 31.

---

### Examples

For examples of the UNLOAD command and UNLOAD output for each entry type, see Chapter 35.

---

### Processing

Before SMP/E unloads any entries, it first determines what type of UNLOAD processing was requested:

- If no specific entries are specified, it does mass-mode processing—for example, if **UNLOAD.** is specified.

Likewise, if specific entry types are specified, SMP/E also does mass-mode processing—for example, if **UNLOAD MAC MOD.** is specified to unload all the MAC and MOD entries in a target zone.

- If specific entries are specified, SMP/E does select-mode processing—for example, if **UNLOAD MAC(MACA,MACB) MOD(MODA,MODB).** is specified to unload specific MAC and MOD entries in a target zone.



**Note:** A single UNLOAD command may combine mass-mode and select-mode processing.

## Mass-Mode Processing

In mass-mode processing, SMP/E checks whether any entry types were specified. If so, SMP/E unloads all entries of the indicated type found in the set-to zone. SMP/E reads sequentially through the set-to zone. For each entry it finds, it checks whether the entry is of the specified type. If the entry meets all the requirements, SMP/E formats and prints the data.

If no entry types were specified, SMP/E unloads all the entries in the set-to zone and reads sequentially through that zone. For each entry it finds, it formats and prints the data.

## Select-Mode Processing

In select-mode processing, SMP/E unloads all the specified entries found in the set-to zone. SMP/E goes directly to each of the entries specified and locates the data for the entry. For each entry it finds, it formats and prints the data.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of UNLOAD processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

**Note:** Either the target zone or the distribution zone is used during setup according to the zone type specified in the previous SET command.

### 2. UNLOAD processing

Global zone	—	Read with shared enqueue.
Target zone	—	Read with shared enqueue.
DLIB zone	—	Read with shared enqueue.

**Note:** The zones used depend on the UNLOAD command operands, and the zone type specified in the previous SET command.

### 3. Termination

All resources are freed.



---

## Chapter 24. The ZONECOPY Command

The ZONECOPY command can be used to copy an entire target or distribution zone from an existing CSI data set to a new CSI data set. With ZONECOPY, you can copy a zone from a CSI containing multiple zones, or from a CSI containing just one zone. SMP/E copies the data from the input zone to the new CSI and renames the receiving zone.

ZONECOPY processing is similar to access method services (AMS) REPRO processing of the same data. To copy a single zone, or to copy a CSI containing just one zone, though, you should not use AMS REPRO—use ZONECOPY instead. In addition to copying the zone, ZONECOPY renames the copied zone for you. However, to copy a complete CSI containing multiple zones, you need to use AMS REPRO. SMP/E does not provide any commands that copy more than one zone at a time.

You can use ZONECOPY to copy the following input zones into the following receiving zones:

- A distribution zone into a distribution zone
- A distribution zone into a target zone
- A target zone into a target zone

**Note:** You cannot copy a target zone into a distribution zone. A global zone cannot be an input or receiving zone.

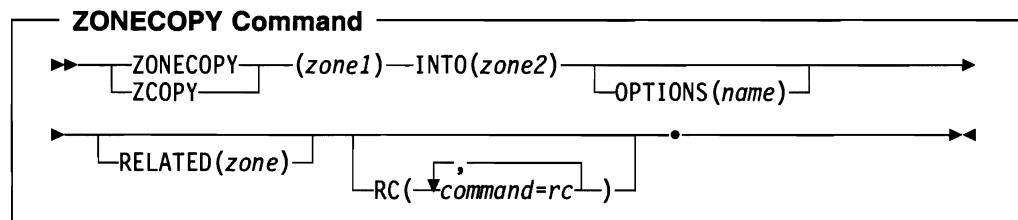
---

### Zones for SET BOUNDARY

For the ZONECOPY command, the SET BOUNDARY command must specify which target or distribution zone should receive the copied zone. This name must match the name of the receiving zone specified on the INTO operand of the ZONECOPY command.

---

### Syntax



---

### Operands

*zone1*

specifies the name of the zone to be copied. This cannot be the global zone.

**INTO**

specifies the name of the zone to receive the copied data. This cannot be the global zone. You can specify the following combinations of input and receiving

zones:

- Distribution zone to distribution zone
- Target zone to target zone
- Distribution zone to target zone

**Note:** A distribution zone cannot receive a target zone.

### OPTIONS

specifies the name of the OPTIONS entry to use for processing the receiving zone. The OPTIONS subentry in the zone definition of the receiving zone is set to the specified name.

### RC

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONECOPY command.

Before SMP/E processes the ZONECOPY command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONECOPY command. Otherwise, the ZONECOPY command fails. For more information about the RC operand, see Appendix D.

### Notes:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONECOPY command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

### RELATED

specifies the name of the target or distribution zone to which the receiving zone is related. The RELATED subentry in the zone definition of the receiving zone is set to the specified name.

## Syntax Notes

The input and receiving zones must meet **all** these conditions:

- Be defined in the same global zone
- Have different names
- Be located in different CSI data sets

If you specify the OPTIONS or RELATED operand, SMP/E either adds new subentries if none exist or replaces the existing subentries with the new ones before writing them to the receiving zone.

---

## Data Sets Used

The following data sets may be needed to run the ZONECOPY command. They **must** be defined by DD statements. They must **not** be defined by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPRPT	<i>zone</i>
SMP_CSI	SMP_OUT	SMP_SNAP	
SMPLOG			

**Note:** *zone* represents the DD statements required for each distribution zone or target zone referred to in this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

## Usage Notes

- Before using the ZONECOPY command, make sure to allocate the new CSI data set and initialize it with a GIMZPOOL record. This data set must be empty except for the GIMZPOOL record. It must never have contained any entries.
- You must define a ZONEINDEX subentry for the receiving zone in the global zone.
- Once you have used ZONECOPY to copy a zone into your new CSI, you cannot use ZONECOPY to copy any additional zones into that CSI. If you plan to use single-zone CSIs, this is not a problem. However, if you want the new CSI to contain multiple zones, you can use other SMP/E commands (such as ZONEMERGE or a combination of ZONEEXPORT and ZONEIMPORT) to copy in additional zones.
- If the copied zone contained any cross-zone subentries, SMP/E issues warning messages. You need to determine what corrective action is needed to ensure that the new zone and any identified cross-zones are properly related through cross-zone subentries. The type of corrective action required depends on:
  - The types of cross-zone subentries found
  - How you plan to use the new zone
  - What you plan to do with the input zone
  - Any actions you plan to perform (or may have already performed) with the zones identified in the warning messages

Here are some examples of possible situations and the associated corrective action:

- The new zone replaces the old zone, which will be deleted.  
You need to update the cross-zone subentries in each identified zone to indicate the new zone name. To do this, use the ZONEEDIT command.
- All the identified zones are being copied to create a clone of the original zone structure.  
You need to connect all the new zones correctly. To do this, use the ZONEEDIT command.
- The new zone contains only TIEDTO and XZMOD subentries, and you want to obtain updates for the identified load modules from the cross-zones.  
You need to update the identified zones with the information required to ensure that the identified modules are automatically linked into the affected load modules. Specifically, you need to add XZLMOD subentries to the

MOD entries in the other zones, as well as a TIEDTO subentry to the TARGETZONE entry for each of these other zones. To add the XZLMOD subentries, use the UCLIN command. To add the TIEDTO subentries, use either the UCLIN command or the SMP/E Administration dialog.

- The new zone is a temporary copy and will be deleted after brief use.

You can either do nothing or disconnect the new zone from all the identified zones. To disconnect the zone, use the ZONEEDIT command to delete all the cross-zone subentries from the new zone.

---

## Output

The File Allocation report is produced during ZONECOPY processing. See Chapter 31 for a description of this report.

---

## Examples

The following examples are provided to help you use the ZONECOPY command:

- “Example 1: Copying a Target Zone to a Target Zone”
- “Example 2: Copying a Distribution Zone to a Distribution Zone” on page 395
- “Example 3: Copying a Distribution Zone to a Target Zone” on page 395

### Example 1: Copying a Target Zone to a Target Zone

Assume you have a test system zone named CPYTEST in data set SMP.TEST.CSI and you need to copy this zone to another target zone, CPYPROD, for a production system. The following SMP/E commands create the new zone:

```
SET          BDY(GLOBAL)      /* Set to global zone.      */.  
UCLIN                /* Define zone index.      */.  
ADD          GZONE           /*                          */.  
            ZINDEX(         /*                          */.  
                (CPYPROD,SMP.PROD.CSI,TARGET) /* */.  
            )              /*                          */.  
ENDUCL                /*                          */.  
SET          BDY(CPYPROD)    /* Set to new zone.        */.  
ZONECOPY    (CPYTEST)       /* Copy target CPYTEST to */.  
            INTO(CPYPROD)   /* target zone CPYPROD.   */.
```

After the ZONECOPY operation, the global zone ZONEINDEX indicates that zone CPYTEST is still in SMP.TEST.CSI and has not been changed. A new global zone ZONEINDEX subentry has been created to indicate that zone CPYPROD is in SMP.PROD.CSI.

**Note:** The two zones must be in different CSI data sets.

## Example 2: Copying a Distribution Zone to a Distribution Zone

Assume you have a distribution zone named TSTDLB1 in data set SMP.DLB1.CSI, with a related target zone TSTTGT1. You want to create a copy of TSTDLB1 to do some testing. The new distribution zone will be named TSTDLB2, will have a related target zone of TSTTGT2, and will be processed according to options entry TSTOPT. The following SMP/E commands create the new zone:

```

SET      BDY(GLOBAL)      /* Set to global zone.      */
UCLIN                                /* Define zone index.      */
ADD      GZONE            /*                          */
          ZINDEX(         /*                          */
            (TSTDLB2,SMP.DLB2.CSI,DLIB) /*                          */
          )                /*                          */
ENDUCL                                /*                          */
SET      BDY(TSTDLB2)     /* Set to new zone.        */
ZONECOPY (TSTDLB1)       /* Copy DLIB zone TSTDLB1 */
          INTO(TSTDLB2)  /* to DLIB zone TSTDLB2.  */
          OPTIONS(TSTOPT) /* Add OPTIONS entry      */
          RELATED(TSTTGT2) /* and related zone.      */

```

After the ZONECOPY operation, the global zone ZONEINDEX indicates that zone TSTDLB1 is still in SMP.DLB1.CSI and has not been changed. A new global zone ZONEINDEX subentry has been created to indicate that zone TSTDLB2 is in SMP.DLB2.CSI, with a new related zone TSTTGT2 and options entry TSTOPT.

**Note:** The two zones must be in different CSI data sets.

## Example 3: Copying a Distribution Zone to a Target Zone

Assume you have a distribution zone named CPYDLIB in data set SMP.DLB.CSI. You have used the distribution libraries described by this zone to do a system generation, and you now want to create a target zone describing the content of your new operating system. The new target zone is to be named CPYTGT and is to be in data set SMP.TGT.CSI. You also want to change the related zone from a target zone to a distribution zone. The following SMP/E commands create the new zone:

```

SET      BDY(GLOBAL)      /* Set to global zone.      */
UCLIN                                /* Define zone index.      */
ADD      GZONE            /*                          */
          ZINDEX(         /*                          */
            (CPYTGT,SMP.TGT.CSI,TARGET) /*                          */
          )                /*                          */
ENDUCL                                /*                          */
SET      BDY(CPYTGT)     /* Set to new zone.        */
ZONECOPY (CPYDLIB)       /* Copy DLIB zone CPYDLIB */
          INTO(CPYTGT)   /* to target zone CPYTGT.  */
          RELATED(CPYDLB1) /* Change related zone.    */

```

After the ZONECOPY operation, the global zone ZONEINDEX specifies that zone CPYDLIB is still in SMP.DLB.CSI and has not been changed. A new global zone ZONEINDEX subentry has been created and specifies that zone CPYTGT is in SMP.TGT.CSI, with a new related zone CPYDLB1.

**Note:** The two zones must be in different CSI data sets.

## Processing

The ZONECOPY command copies a specified distribution or target zone from one CSI data set to another. SMP/E adds the new zone to the end of the receiving CSI data set. The input distribution or target zone remains in the input CSI data set.

Before copying the zone, SMP/E checks that the parameters entered are correct and that the copy request is valid. If any of the checks fail, SMP/E issues an error message, and the ZONECOPY command fails. SMP/E checks that:

- The input and receiving zones are defined in the same global zone.
- The input and receiving zones have different names.
- The input and receiving zones are in different CSI data sets.
- The combination of input and receiving zone types is valid.
- No cross-zone subentry from the input zone refers to a zone with the same name as the receiving zone.

If all the validity checking is successful, the zone can be copied. SMP/E opens the input zone for read access and opens the receiving zone for update processing. SMP/E then reads the data from the input zone and writes the data into the receiving zone.

If the input zone contained cross-zone subentries, SMP/E issues warning messages.

If the input and receiving zone types are different, SMP/E changes the zone type in the zone definition record before writing it to the receiving zone. When it reaches the end of the input zone file, SMP/E closes the input and receiving zones.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of ZONECOPY processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

**Note:** The target zones or distribution zones accessed during this phase are the input zone and the receiving zone.

### 2. ZONECOPY processing

Target zone	—	Read with shared enqueue (input zone).
Target zone	—	Update with exclusive enqueue (receiving zone).
DLIB zone	—	Read with shared enqueue (input zone).
DLIB zone	—	Update with exclusive enqueue (receiving zone).

**Note:** The target or distribution zone specified as the input zone is opened for read access without enqueue, and the target or distribution zone speci-



fied in the INTO operand is opened for update access with exclusive enqueue.

3. Termination

All resources are freed.



---

## Chapter 25. The ZONEDELETE Command

There are times when it is necessary to delete the SMP/E data for one of the systems you are supporting. Examples of when this may become necessary are:

- After performing a full system generation or running the output from the SMP/E GENERATE command, you have to delete the information describing the previous target system libraries and rebuild that information to describe the new set of target system libraries built from the distribution libraries.
- After installing a new level of a product that existed in its own target zone and distribution zone, you want to delete the information about the old level of the product and continue processing only the new level.

The ZONEDELETE command allows you to delete a specified target zone or distribution zone from the CSI in which it was contained.

**Note:** The CSI data set is not deleted; only the specified zone is deleted.

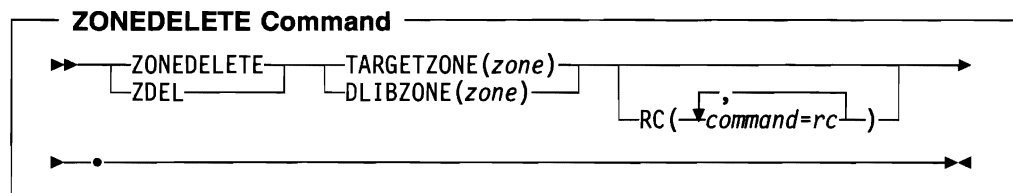
---

### Zones for SET BOUNDARY

For the ZONEDELETE command, the SET BOUNDARY command must specify the name of the zone to be deleted. This same zone must also be specified on the ZONEDELETE command.

---

### Syntax



---

### Operands

#### DLIBZONE

specifies the distribution zone to be deleted.

#### Notes:

1. DLIBZONE can also be specified as DZONE.
2. DLIBZONE is mutually exclusive with TARGETZONE.

#### RC

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONEDELETE command.

Before SMP/E processes the ZONEDELETE command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONEDELETE command. Otherwise, the ZONEDELETE command fails. For more information about the RC operand, see Appendix D.

### Notes:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONEDELETE command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

### TARGETZONE

specifies the target zone to be deleted.

### Notes:

1. TARGETZONE can also be specified as TZONE.
2. TARGETZONE is mutually exclusive with DLIBZONE.

## Syntax Notes

- For SMP/E to determine which CSI data set contains the zone to be deleted, there must already be a zone index for that zone.
- The zone type specified in the zone index must match the type specified on the ZONEDELETE command.

---

## Data Sets Used

The following data sets may be needed to run the ZONEDELETE command. They can be defined by DD statements or, normally, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPRPT	<i>zone</i>
SMP_CSI	SMPOUT	SMPSNAP	
SMPLOG			

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

---

## Usage Notes

- Once a ZONEDELETE command has been successfully processed, the data cannot be recovered unless you have a backup copy of the CSI data set that contained the zone. Therefore, you should be very careful when entering the ZONEDELETE command. Make sure the zone specified is actually the one you want to delete; that is, check the spelling, and so on.
- Once the ZONEDELETE command has been successfully processed, all references to the zone are gone, and nothing more can be done with that zone. Therefore, the next command after ZONEDELETE **must** be a SET command.

- If the deleted zone contained any cross-zone subentries, SMP/E issues warning messages. You need to determine what corrective action is needed to ensure that the other zones have valid cross-zone entries so that SMP/E can function properly. The type of corrective action required depends on:
  - The types of cross-zone subentries found
  - Whether a copy was made of the deleted zone; if so, whether it is being used
  - Any actions you plan to perform (or may have already performed) with the zones identified in the warning messages

Here are some examples of possible situations and the associated corrective action:

- The deleted zone is being removed from the system and will not be replaced.  
You need to delete any cross-zone subentries in the identified zones that refer to the deleted zone. To do this, use the ZONEEDIT command.
- The deleted zone was copied and will be replaced by the copy.  
You need to update the cross-zone subentries in the identified zones with the name of the replacement zone. To do this, use the ZONEEDIT command.

---

## Output

The File Allocation report is produced during ZONDELETE processing. This report is described in Chapter 31.

---

## Examples

The following examples are provided to help you use the ZONDELETE command:

- “Example 1: Deleting a Target Zone”
- “Example 2: Deleting a Distribution Zone” on page 402

### Example 1: Deleting a Target Zone

Assume you have a test system zone named MVSTST1, and you have just performed a full MVS system generation to create a new set of libraries, and then you have done the processing needed to set up a new zone, MVSTST2, representing the new set of target libraries (for examples of setting up a target zone after system generation, see “Examples” on page 426). You are now ready to delete the old target zone:

```
SET      BDY(MVSTST1)      /* Process MVSTST1 tgt zone.*/.
ZDEL     TZONE(MVSTST1)    /* Delete it.                */.
```

## Example 2: Deleting a Distribution Zone

Assume you are running IMS in your installation and that the IMS product exists in its own target zone and distribution zone. You are in the process of migrating from IMS-x to IMS-x+1. IMS-x+1 has been installed into its own distribution zone (that is, not into the distribution zone that IMS-x was in), testing has been completed, and you now want to delete IMS-x. Assume the name of the distribution zone containing IMS-x is IMSX, and the name of the distribution zone containing IMS-x+1 is IMSXP1. The following commands delete the IMS-x distribution zone:

```
SET      BDY(IMSX)          /* Process old IMS zone.   */.
ZDEL     DZONE(IMSX)       /* Delete it.              */.
```

---

## Processing

When a ZONEDELETE command is encountered, SMP/E ensures that you made no mistake in entering the command, as follows:

- SMP/E checks the operand on the ZONEDELETE command (that is, TARGETZONE, TZONE, DLIBZONE, or DZONE) and the zone type specified in the global zone ZONEINDEX to ensure that the zone types match. If they do not, an error condition is reported, and the ZONEDELETE request is not processed.
- SMP/E checks the name specified on the ZONEDELETE command operand to ensure that it matches the names specified in the previous SET command. If not, an error condition is reported, and the ZONEDELETE request is not processed.

If the deleted zone contained cross-zone subentries, SMP/E issues warning messages.

If no verification errors occur, SMP/E deletes the specified zone from the CSI data set it was contained in. After deleting the target zone or distribution zone from the CSI data set, SMP/E deletes the global zone ZONEINDEX entry for that zone. SMP/E now has no references to the deleted zone.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of ZONEDELETE processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

**Note:** Either the target zone or the distribution zone is accessed, according to the zone type specified in the previous SET command.

### 2. Delete zone records

Target zone	—	Update with exclusive enqueue.
DLIB zone	—	Update with exclusive enqueue.

**Note:** Either the target zone or the distribution zone is accessed, according to the zone type specified in the previous SET command.

3. Delete zone index entry

Global zone — Update with exclusive enqueue.

4. Termination

All resources are freed.





---

## Chapter 26. The ZONEEDIT Command

The ZONEEDIT command can be used instead of multiple UCL statements to make mass changes in selected SMP/E entries in the same zone. Here are some examples of when you might want to use the ZONEEDIT command:

- To change all the volume serial numbers in all the DDDEF entries in a specified zone
- To change the ddname for the print output data set in all the UTILITY entries in the global zone
- To change a zone name in all the cross-zone subentries in a specified target zone

---

### Zones for SET BOUNDARY

For the ZONEEDIT command, the SET BOUNDARY command must specify the appropriate type of zone for the entry type that is changed.

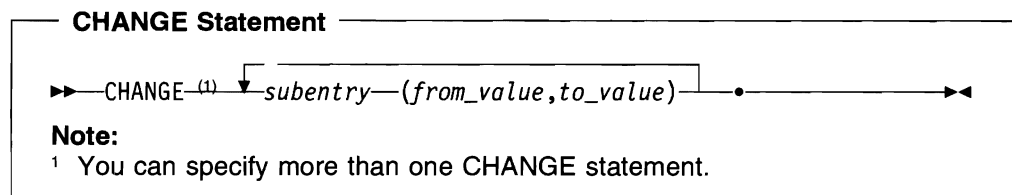
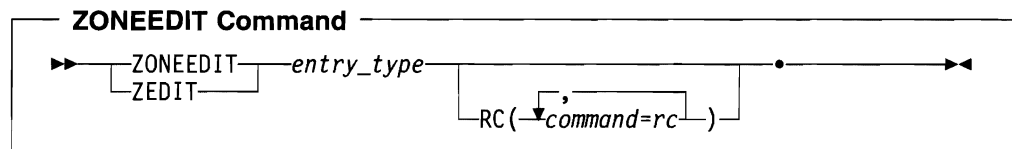
- For DDDEF entries, it must specify the appropriate global, target, or distribution zone.
- For UTILITY entries, it must specify the global zone.
- For cross-zone subentries, it must specify the appropriate target zone.

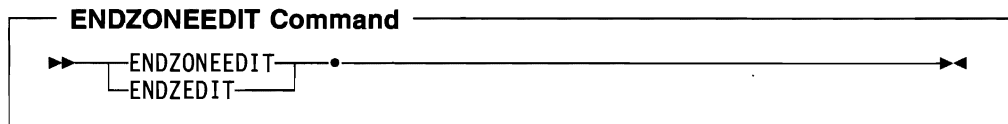
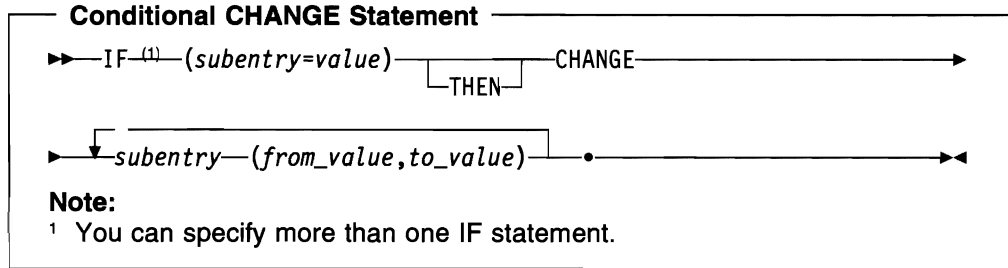
---

### Syntax

Three types of commands are necessary for ZONEEDIT processing:

1. The **ZONEEDIT** command indicates the start of ZONEEDIT processing.
2. ZONEEDIT **CHANGE** statements (for unconditional changes) and **IF...THEN CHANGE** statements (for conditional changes) show the changes to be made.
3. The **ENDZONEEDIT** command indicates the end of the ZONEEDIT commands and the end of ZONEEDIT processing.





## Operands

### *entry-type*

identifies the type of entry to be changed. The valid types are:

- DDDEF (for a global, distribution, or target zone)
- UTILITY (for the global zone only)
- XZENTRIES (for a target zone only)

**Note:** XZENTRIES is not an actual entry type. It is used to change a zone name in all the cross-zone subentries in the specified zone:

- XZLMOD subentries in MOD entries
- XZMOD subentries in LMOD entries
- TIEDTO subentries in the TARGETZONE entry

### **RC**

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONEEDIT command.

Before SMP/E processes the ZONEEDIT command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONEEDIT command. Otherwise, the ZONEEDIT command fails. For more information about the RC operand, see Appendix D.

### **Notes:**

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONEEDIT command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

**CHANGE**

specifies the subentry to be changed, its old value, and its new value.

**Notes:**

1. **CHANGE** can also be specified as **C**.
2. If you specify several subentries on a **CHANGE** statement, use blanks, not commas as separators.

*subentry*

specifies the type of subentry to be changed. These are the values that can be specified:

- For DDDEF entries:  
    DATASET or DA  
    SYSOUT  
    UNIT  
    VOLUME  
    WAITFORDSN or WAIT
- For UTILITY entries:  
    NAME(*utility*)  
    PRINT
- For XZENTRIES:  
    ZONEVALUE

**Note:** ZONEVALUE is not an actual subentry type. It is used to change a zone name in all the cross-zone subentries in the specified zone. You cannot change ZONEVALUE to the name of the set-to zone.

*from-value*

specifies the current value of the subentry. There are three ways to specify this value:

- The actual subentry value.
- \*, which means change the specified subentry regardless of its value. If there is no current value for the subentry, the change is not made.

**Note:** This is not allowed for XZENTRIES.

- *char.\**, where *char* is a character string you specify. This is used for data set names and means SMP/E must change all names beginning with the specified string.

*to-value*

specifies the new value of the subentry. There are three ways to specify this value:

- The actual subentry value.
- \*, which erases the current subentry value.
- *char.\**, where *char* is a character string you specify. This is used for data set names and means SMP/E must change all values meeting the **CHANGE** condition so they begin with the specified string.

### Notes:

1. The recommended method of turning the WAITFORDSN indicator on and off is to use the administration dialogs. However, you can use alternative methods, such as coding **CHANGE WAITFORDSN (YES, NO)** or **CHANGE WAITFORDSN (YES, \*)** to turn the WAITFORDSN indicator off and using UCLIN to turn it on.
2. Coding **CHANGE subentry(\*,\*)** deletes all values for that subentry. You should be sure, therefore, that is what you want when you use this form of the CHANGE statement.

### IF ... THEN

specifies another subentry of the specified entry type that SMP/E is to check before making the change that follows. THEN is optional.

**Note:** Conditional changes are not allowed for XZENTRIES.

## Syntax Notes

- The same subentry name can be specified only once on a single CHANGE command.
- The subentry names must match existing UCLIN subentry names.
- Make sure to specify an existing subentry value. Otherwise, you may get unexpected results.

---

## Data Sets Used

The following data sets may be needed to run the ZONEEDIT command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPRPT	<i>zone</i>
SMP_CSI	SMPOUT	SMPSNAP	
SMPLOG			

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

---

## Output

Two reports are produced during ZONEEDIT processing:

- File Allocation report
- ZONEEDIT Summary report

These reports are described in Chapter 31.

## Examples

The following examples are provided to help you use the ZONEEDIT command:

- “Example 1: Editing DDDEF Entries”
- “Example 2: Conditionally Editing DDDEF Entries”
- “Example 3: Changing the SYSOUT Value”
- “Example 4: Changing the Zone Value in Cross-Zone Subentries” on page 410

### Example 1: Editing DDDEF Entries

In this example, the UNIT value for all the DDDEF entries is to be changed to 3350, regardless of its current value. The following ZONEEDIT command makes this change:

```
SET      BDY (MVSTZ)      /* Set to zone to edit.    */.
ZONEEDIT DDDEF           /* Edit DDDEF entries.    */.
CHANGE   UNIT(*,3350)    /* Change unit to 3350.  */.
ENDZONEEDIT              /* End of ZONEEDIT.      */.
```

### Example 2: Conditionally Editing DDDEF Entries

In this example, only DDDEF entries for data sets on the SYSRES volume are to be changed. UNIT values of 3330 are to be changed to 3350, and DATASET names that start with SYS1 are to start with SYS2. The following ZONEEDIT command makes this change:

```
SET      BDY (MVSTZ)      /* Set to zone to edit.    */.
ZONEEDIT DDDEF           /* Edit DDDEF entries     */.
IF       (VOLUME=SYSRES) /* for SYSRES volume only */
CHANGE   UNIT(3330,3350) /* Change unit from 3330, */
          DATASET(SYS1.*, /* data set name from SYS1. */
          SYS2.*)        /* End of ZONEEDIT.      */.
ENDZONEEDIT              /* End of ZONEEDIT.      */.
```

### Example 3: Changing the SYSOUT Value

In this example, the SYSOUT value for DDDEF entries is to be changed from \* to A. The following ZONEEDIT command makes this change:

```
SET      BDY (MVSTZ)      /* Set to zone to edit.    */.
ZONEEDIT DDDEF           /* Edit DDDEF entries     */.
CHANGE   SYSOUT('* ',A)  /* Change SYSOUT from * to A. */.
ENDZONEEDIT              /* End of ZONEEDIT.      */.
```

There must be quotation marks around the \* if SMP/E is to change only DDDEF entries in which SYSOUT=\*

To change **all** the SYSOUT values to A, the following ZONEEDIT command can be used:

```
SET      BDY (MVSTZ)      /* Set to zone to edit.    */.
ZONEEDIT DDDEF           /* Edit DDDEF entries     */.
CHANGE   SYSOUT(*,A)    /* Change all SYSOUT to A. */.
ENDZONEEDIT              /* End of ZONEEDIT.      */.
```

Because there are no quotation marks around the \*, **all** the SYSOUT values are changed.

## Example 4: Changing the Zone Value in Cross-Zone Subentries

Assume you have used the LINK command to link modules from target zone CICS1 into load modules in target zone MVS1. Later, because of new zone naming conventions, you used the ZONERENAME command to rename zone CICS1 to CICSPRD. During ZONERENAME processing, zone MVS1 was identified as the only zone connected to zone CICS1 by cross-zone subentries.

You now need to change cross-zone subentries in zone MVS1 to show that MVS1 is now connected to the renamed zone, CICSPRD. The following ZONEEDIT command makes this change:

```
SET      BDY (MVS1)      /* Set to zone to edit.      */.
ZONEEDIT XZENTRIES     /* Edit cross-zone subentries.*/
CHANGE  ZONEVALUE(CICS1 /* Change zone from old name */
          CICSPRD)      /* to new name.              */.
ENDZONEEDIT           /* End of ZONEEDIT.         */.
```

---

## Processing

The ZONEEDIT command changes the values for a subentry in different DDDEF or UTILITY entries in the same zone. It also changes the zone name in all the cross-zone subentries in a specified target zone. Before making any changes, SMP/E checks that the entry type specified is DDDEF, UTILITY, or XZENTRIES (for cross-zone subentries). It then opens the specified zone for update.

Next, SMP/E checks that (1) the subentry name is valid for the entry, (2) the subentry name is specified only once on a single CHANGE command, and (3) the *to-value* is valid.

- For any conditional changes, SMP/E tests the IF condition and verifies that the *from-value* exists. If these checks succeed, SMP/E replaces the *from-value* with the *to-value*.
- For unconditional changes, SMP/E verifies that the *from-value* exists and, if so, replaces the *from-value* with the *to-value*.

SMP/E then writes the ZONEEDIT Summary report to the SMPRPT data set and closes the zone.

If any of the checks fail, or if the ENDZONEEDIT command is missing, an error message is issued, and ZONEEDIT processing fails.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of ZONEEDIT processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

**Note:** The global zone, the target zone, or the distribution zone is accessed, according to the zone specified in the previous SET command.

2. ZONEEDIT processing

Global zone	—	Update with exclusive enqueue.
Target zone	—	Update with exclusive enqueue.
DLIB zone	—	Update with exclusive enqueue.

3. Termination

All resources are freed.





---

## Chapter 27. The ZONEEXPORT Command

The ZONEEXPORT command copies a specified zone from a VSAM data set to a sequential data set. There are two ways you can use this copy of the zone:

- As a **backup** copy: You can use this copy to recreate a zone that you or a program erased or otherwise destroyed.
- As a **transportable** copy: You can use this copy to recreate the same zone on another system or in another CSI on the same system.

When you have a backup copy of a zone, you also need backup copies of the SMP/E data sets corresponding to that zone. For SMP/E to restore the zone correctly, these data sets must be synchronized. For example, SMP/E needs these data sets to restore a zone either on another system or as a zone in another CSI on the same system.

The ZONEIMPORT command processes the ZONEEXPORT output to restore the zone. ZONEIMPORT processing is similar to access method services REPRO processing of the same data.

**Note:** Although the access method services REPRO command accepts ZONEEXPORT output, it does not process it correctly. Data in another zone may be replaced, or the new zone may not be accessible to SMP/E. You must, therefore, use the ZONEIMPORT command to process ZONEEXPORT output.

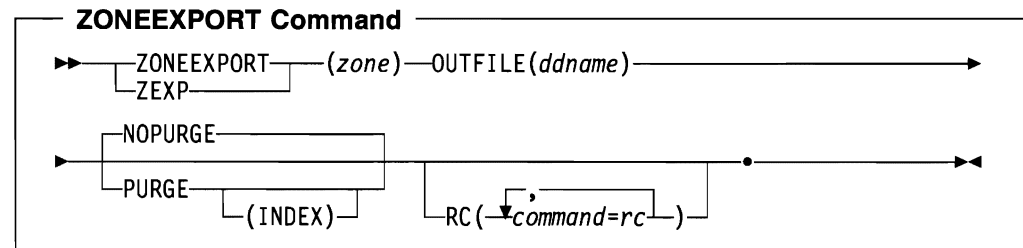
---

### Zones for SET BOUNDARY

For the ZONEEXPORT command, the SET BOUNDARY command must specify the name of the zone to be exported. This name must match the name of the input zone on the ZONEEXPORT command.

---

### Syntax



### Operands

*zone*

specifies the name of the input zone to be copied. This name must match the one specified on the SET BOUNDARY command.

#### OUTFILE

specifies the name of the DD statement that defines the output data set, which is the input for the ZONEIMPORT command.

**Note:** **OUTFILE** can also be specified as **0FILE**.

### **NOPURGE**

specifies that the input zone is **not** to be deleted from the CSI when the ZONEEXPORT is done. This is the default.

**Note:** NOPURGE is mutually exclusive with PURGE.

### **PURGE**

specifies that the input zone is to be deleted from the CSI data set when the ZONEEXPORT is done.

#### **Notes:**

1. PURGE is not allowed for the global zone.
2. PURGE is mutually exclusive with NOPURGE.

### **INDEX**

specifies that the index entry for the input zone is to be deleted from the global zone. If **INDEX** is not specified on the PURGE operand, only the data in the zone is deleted.

### **RC**

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONEEXPORT command.

Before SMP/E processes the ZONEEXPORT command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONEEXPORT command. Otherwise, the ZONEEXPORT command fails. For more information about the RC operand, see Appendix D.

#### **Notes:**

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONEEXPORT command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

---

## Data Sets Used

The following data sets may be needed to run the ZONEEXPORT command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPRPT	<i>zone</i>
SMP_CSI	SMP_OUT	SMP_SNAP	<i>outfile</i>
SMPLOG			

**Notes:**

1. *zone* represents the DD statements required for each distribution zone or target zone referred to in this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.
2. *outfile* represents the DD statement required for the output data set. It is pointed to by the OUTFILE operand.

**Usage Notes**

If the exported zone contained any cross-zone subentries, SMP/E issues warning messages. If you import that zone into a new zone, you need to determine what corrective action is needed to ensure that the new zone and any identified cross-zones are properly related through cross-zone subentries. This corrective action is similar to what you would do after using ZONECOPY. For more information, see “Usage Notes” on page 393.

**Output**

The File Allocation report is produced during ZONEEXPORT processing. It is described in Chapter 31.

**Example: Backing Up Target and Distribution Zones**

In this example, two zones are exported. The first, a target zone named MVSTZ, is backed up and deleted. The second, a distribution zone named MVSDZ, is backed up but is not deleted.

```

SET          BDY (MVSTZ)          /* Set to zone to export. */.
ZONEEXPORT  (MVSTZ)              /* Export MVSTZ. */.
           OUTFILE(EXPORT1) /* DD statement for output. */.
           PURGE                 /* Delete MVSTZ. */.
SET          BDY (MVSDZ)          /* Set to zone to export. */.
ZONEEXPORT  (MVSDZ)              /* Export MVSDZ. */.
           OFILE(EXPORT2)      /* DD statement for output. */.
           NOPURGE              /* Do not delete MVSDZ. */.
    
```

**Processing**

The ZONEEXPORT command makes a backup or transportable copy of a specified distribution, target, or global zone.

Before doing the export, SMP/E checks that the correct parameters were entered and that the export request is valid. If any of the checks fail, SMP/E issues an error message, and the ZONEEXPORT command fails. SMP/E checks that:

- The zone name on the SET BOUNDARY command matches the input zone name.
- If the zone to be exported is the global zone, the PURGE operand is not specified.

If all the validity checking is successful, the zone can be exported. SMP/E opens the input zone and reads the first record. If the input zone is not the global zone, this first record is the zone definition record. (For the global zone, SMP/E must generate a zone definition record.)

SMP/E then opens the output data set and writes the zone definition record to the output data set. Next, SMP/E reads the data from the input zone and writes it to the output data set using sequential reads and writes. After reading all the records from the input zone, SMP/E writes a record containing hex FFFF to the output data set. This record defines the end of data for a successful zone export.

SMP/E then closes the output data set and writes a message to indicate that the data has been written out successfully. If **PURGE** was specified, the records in the zone are deleted by **ZONEDELETE**. If **INDEX** was specified on the **PURGE** operand, and the global zone is available, the associated zone index is also deleted. Otherwise, the zone index is not changed. Finally, SMP/E closes the input zone.

If the exported zone contained cross-zone subentries, SMP/E issues warning messages. If SMP/E cannot open the input data set or the receiving zone, it issues an error message, and the **ZONEEXPORT** command fails.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of **ZONEEXPORT** processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

**Note:** The global zone and the zone specified in the **SET BOUNDARY** command are opened for read access only.

### 2. Zone export processing

Global zone	—	Read with shared enqueue.
-------------	---	---------------------------

If **PURGE(INDEX)** is specified, this is opened for update with exclusive enqueue.

Target zone	—	Read with shared enqueue.
-------------	---	---------------------------

If **PURGE** is specified, this is opened for update with exclusive enqueue.

DLIB zone	—	Read with shared enqueue.
-----------	---	---------------------------

If **PURGE** is specified, this is opened for update with exclusive enqueue.

**Note:** The zone specified in the previous **SET** command is accessed.

### 3. Termination

All resources are freed.

---

## Chapter 28. The ZONEIMPORT Command

The ZONEIMPORT command loads an exported zone from a sequential data set into a specified distribution, target, or global zone. If you are importing a target or distribution zone, you can also change the zone name. (You cannot change the name of a global zone that is being imported.) Generally, you can import a zone only into the same type of zone. However, you can import a distribution zone into a target zone. You might want to do this after system generation, when the target zone is being initialized.

The input data for the ZONEIMPORT command must be the output from a ZONEEXPORT command. ZONEIMPORT processing of this data is similar to Access Method Services REPRO processing of this same data.

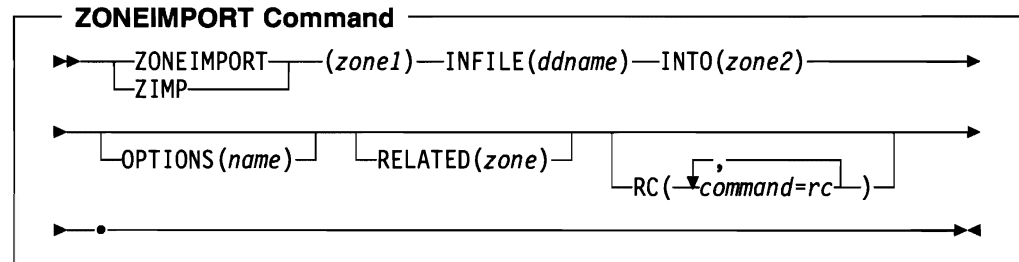
---

### Zones for SET BOUNDARY

For the ZONEIMPORT command, the SET BOUNDARY command must specify where to load the zone. This name must match the name of the receiving zone specified on the INTO operand of the ZONEIMPORT command.

---

### Syntax



### Operands

*zone1*

specifies the name of the input zone to be imported. This is the name of the zone that was exported.

**Notes:**

1. If you are importing a global zone, you must specify **GLOBAL**.
2. If you are importing a target or distribution zone, you can keep the same zone name or rename the zone:
  - If you are keeping the same zone name, you must specify the same zone here and on the SET BOUNDARY command.
  - If you are renaming the zone, this zone name is different from the one specified on the SET BOUNDARY command.

### INFILE

specifies the name of the DD statement that defines the input data set created by the ZONEEXPORT command. You can import only a data set that was created by the ZONEEXPORT command.

**Note:** INFILE can also be specified as IFILE.

### INTO

specifies the name of the receiving zone. This must match the name specified on the SET BOUNDARY command. If the INTO operand specifies a global zone, or you are importing a target or distribution zone to a new CSI data set, you should make sure you have allocated the CSI data set before you run the ZONEIMPORT command, and initialize it with a GIMZPOOL record. **Do not** add any other entries or subentries to the zone.

**Note:** If the INTO operand specifies a target or distribution zone, the global zone must already contain a ZONEINDEX subentry specifying the name of that zone and the name of the CSI data set containing that zone.

### OPTIONS

specifies the name of the OPTIONS entry to use for processing the receiving zone. The OPTIONS subentry in the zone definition of the receiving zone is set to the specified name.

### RC

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONEIMPORT command.

Before SMP/E processes the ZONEIMPORT command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONEIMPORT command. Otherwise, the ZONEIMPORT command fails. For more information about the RC operand, see Appendix D.

#### Notes:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONEIMPORT command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

### RELATED

specifies the name of the target or distribution zone to which the receiving zone is related. The RELATED subentry in the zone definition of the receiving zone is set to the specified name.

---

## Data Sets Used

The following data sets may be needed to run the ZONEIMPORT command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPRPT	<i>zone</i>
SMP_CSI	SMP_OUT	SMP_SNAP	<i>infile</i>
SMPLOG			

### Notes:

1. *zone* represents the DD statements required for each distribution zone or target zone referred to in this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.
2. *infile* represents the DD statement required for the input data set. It is pointed to by the INFILE operand.

---

## Usage Notes

If the imported zone contains any cross-zone subentries, SMP/E issues warning messages. You need to determine what corrective action is needed to ensure that the imported zone and any identified cross-zones are properly related through cross-zone subentries. This corrective action is similar to what you would do after using ZONECOPY. For more information, see “Usage Notes” on page 393.

---

## Output

The File Allocation report is produced during ZONEIMPORT processing. It is described in Chapter 31.

---

## Examples

The following examples are provided to help you use the ZONEIMPORT command:

- “Example 1: Importing a Distribution Zone into a Target Zone” on page 420
- “Example 2: Importing a Global Zone” on page 420

## Example 1: Importing a Distribution Zone into a Target Zone

Assume you have exported a distribution zone named IMPDLB into file IMPFILE, and now you want to use this zone to create a target zone named IMPTGT in an existing data set, SMP.IMPORT.CSI. You need an IMPFILE DD statement to point to the exported file, and an SMPCSI DD statement to point to the CSI data set that contains the global zone. The following SMP/E commands create the new zone:

```

SET      BDY(GLOBAL)      /* Set to global zone.      */.
UCLIN                                /* Define zone index.      */.
ADD      GZONE            /*                          */.
          ZINDEX(         /*                          */.
              (IMPTGT,SMP.IMPORT.CSI,TARGET) /* */.
              )           /*                          */.
ENDUCL                                /*                          */.
SET      BDY(IMPTGT)      /* Set to new zone.        */.
ZIMP     (IMPDLB)         /* Import DLIB zone IMPDLB */.
          INFILE(IMPFILE) /* into target zone IMPTGT.*/.
          INTO(IMPTGT)    /*                          */.
          RELATED(IMPDLB1) /* Add related zone.      */.

```

After the ZONEIMPORT operation, a new global zone ZONEINDEX entry has been created to indicate that zone IMPTGT is in SMP.IMPORT.CSI and has a related zone, IMPDLB1. Note that you changed both the name and type of the zone when you imported it.

## Example 2: Importing a Global Zone

Assume you need to copy your global zone into another system for testing. You must first use ZONEEXPORT to unload the global zone data to a file (see Chapter 27 for more information). Then, you must allocate a new CSI data set and initialize it with only a ZPOOL record. There must not be any other records in this new data set.

Assume you exported the global zone to a file, whose ddname is GOUTFILE, and you allocated a new CSI data set, called SMP.IMPORT.GLOBAL.CSI, for the global zone. You need a GOUTFILE DD statement to point to the exported file and an SMPCSI DD statement to point to the new CSI data set, where you are importing the global zone. The following SMP/E commands import the global zone:

```

SET      BDY(GLOBAL)      /* Set to global zone.      */.
ZONEIMPORT (GLOBAL)      /* Import the global zone.  */.
          INFILE(GOUTFILE) /*                          */.
          INTO(GLOBAL)    /*                          */.

```

After the ZONEIMPORT operation, you have a new global zone in SMP.IMPORT.GLOBAL.CSI. The exported data remains in file GOUTFILE.

## Processing

The ZONEIMPORT command loads an exported zone into a specified distribution, target, or global zone.



Before importing the zone, SMP/E checks that the correct parameters were entered and that the import request is valid. If any of the checks fail, SMP/E issues an error message, and the ZONEIMPORT command fails. SMP/E checks that:

- The zone name on the SET BOUNDARY command matches the receiving zone name.
- Either the zone types of the exported and receiving zones match, or a distribution zone is being loaded into a target zone.
- The receiving zone does not already exist in the CSI data set.
- If the receiving zone is a global zone, no other zone is defined in the CSI.
- No cross-zone subentry from the input zone refers to a zone with the same name as the receiving zone.

If all the validity checking is successful, the zone can be imported. SMP/E opens the input data set, reads the zone definition record, and checks its validity. SMP/E opens the receiving zone for update processing. If a target or distribution zone is being imported and its zone name does not match the name of the receiving zone, SMP/E renames the zone being imported.

SMP/E checks the following:

- If the imported zone contains cross-zone subentries, SMP/E issues warning messages.
- If SMP/E cannot open the input data set or the receiving zone, it issues an error message, and the ZONEIMPORT command fails.
- If the zone name or type was changed, SMP/E updates the zone definition record. If the receiving zone is not the global zone, SMP/E writes this record to the receiving zone. SMP/E then reads the rest of the records from the input data set and writes them to the receiving zone using sequential reads and writes. When it reaches the record containing hex FFFF, SMP/E has finished importing the zone. It does not write this record to the receiving zone.
- If SMP/E reaches the end of the file before reading the hex FFFF record, it issues an error message and the ZONEIMPORT command fails.
- If you specified the OPTIONS or RELATED operand, SMP/E either writes the new entries to the receiving zone, if none already exist, or replaces the existing entries with the new ones and writes the new entries to the receiving zone.

SMP/E then closes the input data set and the receiving zone.

---

## Zone and Data Set Sharing Considerations

The following identifies the phase of ZONEIMPORT processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

**Note:** The zone specified in the SET command is accessed.

### 2. Zone import processing

Global zone	—	Update with exclusive enqueue.
Target zone	—	Update with exclusive enqueue.
DLIB zone	—	Update with exclusive enqueue.

**Note:** The zone specified in the SET command is accessed.

### 3. Termination

All resources are freed.

---

## Chapter 29. The ZONEMERGE Command

The ZONEMERGE command can be used to:

- **Copy** one zone to another (that is, merge an existing zone into a null zone). For example, you can use ZONEMERGE to:
  - Copy a distribution zone to a new target zone after a full system generation.
  - Prime a new distribution zone or target zone with data from an existing distribution zone or target zone before you build a new system.
- **Merge** two existing zones together.

This function of ZONEMERGE should be used with caution. It should be used only to merge two zones that have absolutely no intersections (such as no common modules, macros, source, load modules, or SYSMODs). For further cautions on using ZONEMERGE to merge zones, see “Usage Notes” on page 425.

### Notes:

1. The zones processed by the ZONEMERGE command can either be in the same CSI data set or in different ones.
2. The ZONEMERGE command is the only method you should use to merge zones within the same CSI data set.
3. **Do not use the access method services REPRO command to merge CSI data sets.**

Each CSI data set contains special records that record the names of the zones within that CSI, and an encoded value for that name. That encoded value is stored in each of the records in the CSI associated with that zone. Multiple CSI data sets can have the same encoded value for different zones. When ZONEMERGE is used to merge zones of a CSI, SMP/E can resolve the encoded names and assign new values to the data in each record. When access method services REPRO is used, however, the data is merged together; the result is an unusable CSI data set.

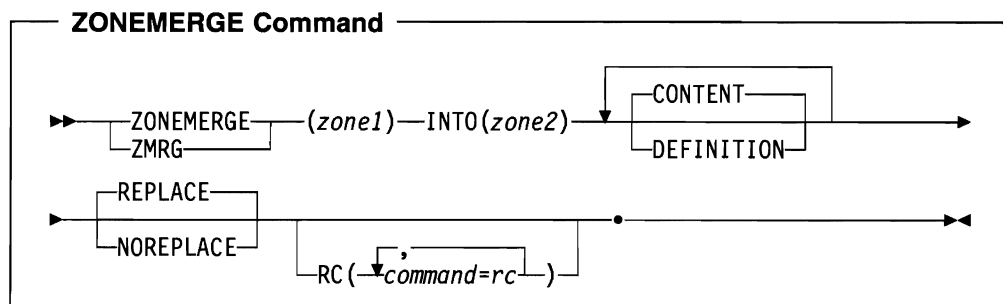
4. ZONEMERGE merges data only in CSI data sets. It does not affect the target or distribution libraries. After using the ZONEMERGE command, therefore, you must ensure that the target library or the distribution library is coordinated with the resulting target or distribution zone.

---

### Zones for SET BOUNDARY

For the ZONEMERGE command, the SET BOUNDARY command must specify the name of the target zone or distribution zone into which the data is to be merged. This name must match the name of the zone specified on the INTO operand of the ZONEMERGE command.

## Syntax



## Operands

### *zone1*

specifies the name of the zone from which the data to be merged is to be obtained (the originating zone).

### **CONTENT**

specifies that the content entries in the zone should be copied. Content entries include SYSMOD entries, element entries, and LMOD entries. If neither **CONTENT** nor **DEFINITION** is specified, **CONTENT** is the default.

**Note:** **CONTENT** can also be specified as **CON**.

### **DEFINITION**

specifies that the DDDEF entries in the zone should be copied.

**Note:** **DEFINITION** can also be specified as **DEF**.

### **INTO**

specifies the zone into which the data from the originating zone is merged (the destination zone). This operand is required.

### **NOREPLACE**

specifies that if the destination zone contains the same entries as the originating zone, the originating zone entries **are not** to overlay the destination zone entries.

**Note:** **NOREPLACE** is mutually exclusive with **REPLACE**.

### **RC**

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONEMERGE command.

Before SMP/E processes the ZONEMERGE command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONEMERGE command. Otherwise, the ZONEMERGE command fails. For more information about the RC operand, see Appendix D.

**Notes:**

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONEMERGE command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

**REPLACE**

specifies that if the destination zone contains the same entries as the originating zone, the originating zone entries **are** to overlay the destination zone entries. This is the default.

**Note:** REPLACE is mutually exclusive with NOREPLACE.

---

**Data Sets Used**

The following data sets may be needed to run the ZONEMERGE command. They may be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPRPT	<i>zone</i>
SMP_CSI	SMP_OUT	SMP_SNAP	
SMPLOG			

**Note:** *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. A DD statement is required for both the originating zone and the destination zone.

---

**Usage Notes**

- Use the ZONEMERGE command with caution. You can get unexpected results if the zones being merged have entries with the same name, or if you mistakenly specified **REPLACE** instead of **NOREPLACE** (or vice versa). For example, suppose both ZONE1 and ZONE2 contain an entry for module IFBMOD01. In ZONE1, the RMID is UZ12345, and in ZONE2, the RMID is UZ12346. Assume UZ12346 is at a higher service level than UZ12345. If ZONE2 were merged into ZONE1 with the REPLACE option, but the target library still contained the version of the module from UZ12345, the resulting entry for module IFBMOD01 in ZONE1 would reflect the incorrect service level of the module.
- If the originating zone contains any cross-zone subentries, SMP/E issues warning messages. You need to determine what corrective action is needed to ensure that the cross-zone information for the merged entries and the identified cross-zones is correct, and that all zones are properly connected. This corrective action is similar to what you would do after using ZONECOPY. For more information, see “Usage Notes” on page 393.

---

## Output

Two reports are produced during ZONEMERGE processing:

- File Allocation report
- ZONEMERGE report

These reports are described in Chapter 31.

---

## Examples

The following examples are provided to help you use the ZONEMERGE command:

- “Example 1: Creating New Target Zone after System Generation”
- “Example 2: Creating a Test Target System” on page 428
- “Example 3: Creating a Test Distribution System” on page 429

**Note:** For most cases where ZONEMERGE can be used to copy zones, you can also use the access method services REPRO command and ZONERENAME to achieve the same result. For examples of ZONERENAME, see “Examples” on page 437.

### Example 1: Creating New Target Zone after System Generation

After performing a full system generation, you need to create a new target zone describing the new set of target libraries created during the system generation process. The following example illustrates the steps necessary to do this.

The first step is to define the new target zone. Assume the previous target zone was named TGT1, the new target zone is to be named TGT2 (generated from distribution zone DLB1), and the OPTIONS entry to be used is MVSOPT. Both target zones are to exist in SMPE.SMPCSI.CSI. The following SMP/E commands define the new target zone:

```

SET      BDY(GLOBAL)          /* Set to global.          */.
UCLIN    /* UCLIN to define new zone.*/.
ADD      GZONE                /*                            */.
          ZONEINDEX(          /* Define zone in CSI.     */.
              (TGT2,SMPE.SMPCSI.CSI,TARGET) /*                            */.
          )                    /*                            */.
          /* Define new zone.          */.
ENDUCL   /*                            */.
SET      BDY(TGT2)            /* Set to new zone.        */.
UCLIN    /* Add definition entry.      */.
ADD      TZONE(TGT2)         /* Define zone.            */.
          RELATED(DLB1)       /* Same info as            */.
          OPTIONS(MVSOPT)     /* in old zone.           */.
          SREL(Z038)          /*                            */.
ENDUCL   /*                            */.

```

Now that the new target zone has been defined, you should use the ZONEMERGE command to copy the entries from the old target zone that do not reflect system structure information. This can be done by using the DEFINITION operand of the ZONEMERGE command. The only entries that are copied in this case are the DDDEF entries, and these should not have changed during the system generation

process. The following SMP/E commands copy the DDDEF entries:

```
SET      BDY(TGT2)          /* Set to new zone.      */.
ZONEMERGE(TGT1)          /* Merge from TGT1      */.
        INTO(TGT2)        /* into new zone TGT2.  */.
        DEFINITION        /* Definition only.     */.
```

**Note:** If you have changed the unit or VOLSER of the target volumes, and the DDDEF entries describing those libraries also contains the unit or volume information (that is, the DDDEF entries did not assume that the data sets were cataloged), these entries must be modified with UCLIN to reflect the new data. The following is an example of how to modify the DDDEF entries to change both the unit and volume information (assume old unit and volume were 3330 and TGTVOL and new information is 3380 and TGTPCK):

```
SET      BDY(TGT2)          /* Set to new target zone. */.
UCLIN                                         /*                          */.
REP      DDDEF(MACLIB)      /* MACLIB changed to      */.
        UNIT(3380)         /* unit 3380,              */.
        VOLUME(TGTPCK)     /* volume TGTPCK.         */.
                                         /* dsname not changed.    */.
ENDUCL                                       /*                          */.
```

At this point, you must copy the distribution zone to the new target zone so SMP/E can determine the function and service levels of all of the distribution library elements. This copy can be done as follows:

```
SET      BDY(TGT2)          /* Set to new target zone. */.
ZONEMERGE(DLB1)          /* Copy from DLIB         */.
        INTO(TGT2)        /* into new target system, */.
        CONTENT           /* but only the element   */.
                                         /* and SYSMOD entries.    */.
```

Now that the target zone is defined and primed with the nonstructure entries from the previous target zone, you now have to prime the new target zone with the structure information about the entries in the new target libraries. You can do this with the JCLIN command of SMP/E, using the stage 1 output as input. Assuming the stage 1 output was saved in data set STG1.TGT2.CNTL and an SMP/E procedure similar to SMPPROC, as described in Appendix B, is available, the following job primes the new target zone:

```
//JOB1      JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1     EXEC SMPPROC
//SMP.SMPJCLIN DD DSN=STG1.TGT2.CNTL,DISP=SHR
//SYSIN     DD *
SET      BDY(TGT2)          /* Set to new target.     */.
JCLIN                                         /* Update zone.          */.
LIST                                           /* List zone.            */.
/*
```

The new target zone, TGT2, is now ready to be used for the installation of service or new function. When this system has been tested and the old level, TGT1, is no longer required, you can delete it by use of the ZONEDELETE command, as follows:

```
SET      BDY(TGT1)          /* Set to old target.    */.
ZONEDELETE          /*                               */.
          TZONE(TGT1)      /* Delete it.           */.
```

## Example 2: Creating a Test Target System

Assume you have a target zone named TSTTGT1, and that you want to create a backup copy of that zone, to be named BKPTGT1, before installing a new product. For this example, assume the libraries themselves have already been backed up. Also assume the CSI describing the TSTTGT1 target zone is SMPE.SMPCSI.CSI, and the CSI that will contain the backup zone BKPTGT1 is SYS1.BACKUP.SMPCSI.CSI. The following set of commands creates the backup zone:

```
SET      BDY(GLOBAL)       /* Set to global.       */.
UCLIN                               /* UCLIN to define new zone.*/.
ADD      GZONE             /*                               */.
          ZONEINDEX(       /* Define zone in CSI.   */.
              (BKPTGT1,SYS1.BACKUP.SMPCSI.CSI,TARGET)
          )                /*                               */.
                               /* Define new zone.     */.
ENDUCL                               /*                               */.
SET      BDY(BKPTGT1)     /* Set to new zone.     */.
UCLIN                               /* Add definition entry. */.
ADD      TZONE(BKPTGT1)   /* Define zone.         */.
          RELATED(TSTDLB1) /* Same info as         */.
          OPTIONS(MVSOPT1) /* in old zone.        */.
          SREL(Z038)      /*                               */.
ENDUCL                               /*                               */.
ZONEMERGE(TSTTGT1)       /* Merge(copy) TSTTGT1  */.
          INTO(BKPTGT1)   /* into new zone,      */.
          CONTENT        /* all type entries.   */.
          DEFINITION     /*                               */.
                               /* Replace/noreplace is not
                               necessary -- destination
                               zone is empty.          */.
LIST                               /* List the new zone.   */.
```



### Example 3: Creating a Test Distribution System

Assume you have a distribution zone named TSTDLB1, and you want to make a copy of it, to be named TSTDLB2, for use in some sort of testing. You first have to make copies of the libraries associated with that distribution zone. After doing that, you can use SMP/E to make a copy of the distribution zone. You can use the following set of commands to do this:

```

SET      BDY(GLOBAL)          /* Set to global.          */
UCLIN                                /* UCLIN to define new zone.*/
ADD      GZONE                /*                          */
          ZONEINDEX(          /* Define zone in CSI.     */
              (TSTDLB2,SMPE.SMPCSI.CSI,DLIB) /*
              )              /*                          */
          )                    /* Define new zone.       */
ENDUCL                                /*                          */
SET      BDY(TSTDLB2)         /* Set to new zone.       */
UCLIN                                /* Add definition entry.   */
ADD      DZONE(TSTDLB2)      /* Define zone.           */
          RELATED(TSTTGT1)    /* Same info as           */
          OPTIONS(MVSOPT1)    /* in old zone.          */
          SREL(Z038)          /*                          */
ENDUCL                                /*                          */
ZONEMERGE(TSTDLB1)          /* Merge(copy) TSTDLB1   */
          INTO(TSTDLB2)       /* into new zone,        */
          CONTENT             /* all type entries.    */
          DEFINITION          /*                          */
          /* Replace/noreplace is not
          necessary -- destination
          zone is empty.     */
LIST                                /* List the new zone.    */

```

### Processing

The ZONEMERGE command allows you to merge a specified distribution zone or target zone into another specified target zone or distribution zone. REPLACE is the default if neither REPLACE nor NOREPLACE is specified. After the merge operation is complete, the originating distribution zone or target zone still exists in the CSI data set.

SMP/E supports the following variations of merging:

- Distribution zone to distribution zone
- Target zone to target zone
- Distribution zone to target zone

**Note:** Any attempt to merge a target zone to a distribution zone causes an error.

The syntax refers to two types of zone entries: **DEFINITION** and **CONTENT**. **DEFINITION** entries are set up by the user to define data sets used by SMP/E; **CONTENT** entries are created by SMP/E. If neither **CONTENT** nor **DEFINITION** is specified, the default is to merge only the **CONTENT** type entries. This supports the distribution zone to target zone copy after a system generation when the desire is to recopy the structure of the system, but not the definition. The following defines how the various entries are categorized:

- **DEFINITION** type entries
  - **DDDEF** entries
- **CONTENT** type entries
  - **ASSEM** entries
  - **Element** entries
  - **DLIB** entries
  - **LMOD** entries
  - **SYSMOD** entries

When the **ZONEMERGE** command is encountered, SMP/E first ensures that both zones have been previously defined. The **ZONEMERGE** command does not create a new zone. If either one has not been defined, an error message is issued, and the command is not processed. To fix this problem, check the zone names specified, and either change the incorrect name, or define the missing zone by using **UCLIN**.

SMP/E then checks to make sure the zone type (that is, target or distribution), specified in the global zone **ZONEINDEX**, matches the type specified in the **ZONEMERGE** command. If not, an error message is issued, and the **ZONEMERGE** is not done. To fix this problem, check to ensure that the zone names specified are correct and that the zone types specified in the global zone **ZONEINDEX** are correct; fix whatever discrepancy was found, and rerun the command.

To perform the actual merge, SMP/E sequentially reads through both zones. For each entry in the originating zone, SMP/E checks the entry type (**DEFINITION**, **CONTENT**, or both) and looks at the operands specified on the **ZONEMERGE** command to determine whether the entry should be processed. If so, SMP/E checks to see whether that entry exists in the destination zone.

- If an entry is not found, the entry from the originating zone is stored.
- If an entry is found and **REPLACE** was specified, the entry from the destination zone is deleted, and the entry from the originating zone is stored.
- If an entry is found and **NOREPLACE** was specified, processing continues with the next entry.

If the originating zone contained cross-zone subentries, SMP/E issues warning messages. SMP/E determines what cross-zone information to merge from **MOD** and **LMOD** entries in the originating zone, as follows:

- If any cross-zone subentries refer to a zone with the same name as the receiving zone, that information is not copied to the receiving zone, and an error message is issued. SMP/E still attempts to merge the rest of the data in the zone.

- If the entry in the originating zone contains nothing but cross-zone subentries, SMP/E does not copy the cross-zone subentries to the receiving zone.
- If the entry in the receiving zone contains nothing but cross-zone subentries, SMP/E merges the entry from the originating zone into the receiving zone, regardless of whether the REPLACE operand was specified. Cross-zone subentries from the originating zone are added to those from the receiving zone. SMP/E issues messages indicating the source of each of the cross-zone subentries in the merged entry.
- Otherwise, the entries are merged only if REPLACE was specified. In this case, SMP/E merges the entry from the originating zone into the receiving zone, replacing all the information in the receiving zone (except the cross-zone subentries) with information from the originating zone. Cross-zone subentries from the originating zone are added to those from the receiving zone. SMP/E issues messages to indicate the source of each of the cross-zone subentries in the merged entry.
- If any cross-zone subentries were merged into the receiving zone, SMP/E adds TIEDTO subentries for the cross-zones to the receiving zone's TARGETZONE entry.

When you merge zones, you can end up with an LMOD entry with a MODDEL subentry for a module and a MOD entry for that same module. For example, suppose you have a zone with an LMOD entry that has a MODDEL subentry containing module MOD005, and you merge that zone with another zone that has a MOD entry for MOD005. It might seem contradictory, but the result is a zone having an LMOD entry with a MODDEL subentry containing MOD005 and a MOD entry for MOD005. If you install a SYSMOD containing MOD005, module MOD005 is removed from the MODDEL subentry.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of ZONEMERGE processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

#### Notes:

- Either the target zone or the distribution zone is accessed, according to the zone type specified in the previous SET command.
- Both the “from” and “to” zones are accessed at this time.

### 2. ZONEMERGE processing

Target zone	—	Read with shared enqueue.
Target zone	—	Update with exclusive enqueue.
DLIB zone	—	Read with shared enqueue.
DLIB zone	—	Update with exclusive enqueue.

**Notes:**

- a. Either the target zone or the distribution zone is accessed, according to the zone type specified in the previous SET command.
- b. The “from” zone is accessed for read with shared enqueue; the “to” zone is accessed for update with exclusive enqueue.

3. Termination

All resources are freed.

---

## Chapter 30. The ZONERENAME Command

The ZONERENAME command allows you to assign a new name to an existing target zone or distribution zone. It also allows you to change the zone type from DLIB to TARGET, change the default OPTIONS entry name, and connect a target zone to a different related distribution zone.

You may want to use the ZONERENAME command in some of the following situations:

- If the current name of a zone does not conform to naming conventions of the establishment controlling the zone, you want to assign a new name to the zone without going through the overhead required to do a ZONEMERGE followed by a ZONEDELETE.
- If you create a new set of target libraries during system generation and need a target zone to describe them, you have two choices. You can use the ZONEMERGE command to create the target zone (see "Examples" on page 426 for further information), or you can make a copy of the CSI containing the distribution zone. Then, using the ZONERENAME command, you can rename the distribution zone in the CSI copy to your new target zone name, change the zone type to TARGET, and connect it to the original distribution zone.
- If you have made a copy of a CSI data set and want to access, through your master global zone, one of the zones in the copy. Assuming that the zones in the original CSI are already accessed through that global zone, you cannot also access the zones in the copy, because there can be only one ZONEINDEX entry in the global zone for each unique zone name.

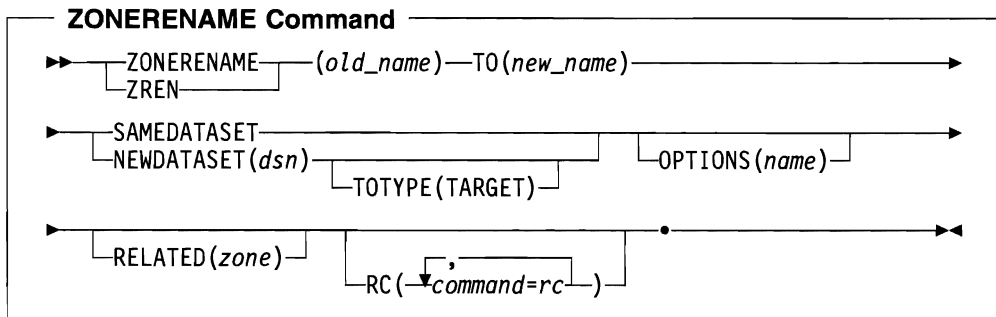
In this case, you want to rename the zone in the CSI copy, and then connect the renamed zone in your global zone. This procedure can be used to make a working copy of a zone under a different name for testing purposes without the overhead required to do a ZONEMERGE.

---

### Zones for SET BOUNDARY

For the ZONERENAME command, the SET BOUNDARY command must specify the global zone.

## Syntax



## Operands

### *old-name*

specifies the name of the zone to be renamed.

#### Notes:

1. The specified name cannot be **GLOBAL**.
2. The old zone name cannot be the same as the new zone name.

### **TO**

specifies the new name for the zone. This operand is required.

#### Notes:

1. The new zone name cannot be **GLOBAL**.
2. The new zone name cannot be the same as the old zone name.

### **NEWDATASET**

indicates that the new zone name is not yet defined by a zone index in the global zone. However, the old zone does exist in the indicated data set. SMP/E is to rename the zone in the indicated data set, and then create a zone index for the new zone name.

You must specify either **NEWDATASET** or **SAMEDATASET**.

#### Notes:

1. The data set must not be the same as the one currently defined in the zone index for the old zone name.
2. The data set name must conform to the naming conventions for a CSI data set, and, therefore, must have a low-level qualifier of **.CSI**. The data set name can contain no more than 44 characters.

### **OPTIONS**

specifies the name of the **OPTIONS** entry to use for processing the renamed zone. The **OPTIONS** subentry in the zone definition of the **RENAMED** zone is set to the specified name. If **OPTIONS** is not specified, the **OPTIONS** subentry in the definition of the renamed zone is not changed.

### **RC**

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the **ZONERENAME** command.

Before SMP/E processes the ZONERENAME command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONERENAME command. Otherwise, the ZONERENAME command fails. For more information about the RC operand, see Appendix D.

**Notes:**

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONERENAME command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

**RELATED**

specifies the name of the target or distribution zone to which the renamed zone is related. The RELATED subentry in the zone definition of the renamed zone is set to the specified name. If RELATED is not specified, the RELATED subentry in the renamed zone's definition is not changed.

**SAMEDATASET**

indicates that the zone to be renamed is already defined by a zone index in the global zone, and is to be renamed in its current data set. The zone index is be changed to indicate the new zone name.

You must specify either SAMEDATASET or NEWDATASET.

**Note:** SAMEDATASET is mutually exclusive with TOTYPE(TARGET).

**TOTYPE(TARGET)**

indicates that the zone to be renamed is currently a distribution zone in a copy of an SMP/E CSI data set and is to be changed to a target zone. After the rename operation, the GLOBALZONE ZONEINDEX record indicates that this is now a target zone and the zone definition for the zone is updated to indicate that it is a target zone.

**Notes:**

1. TOTYPE is mutually exclusive with SAMEDATASET.
2. TOTYPE is needed only if you are changing a target zone to a distribution zone. For example, suppose you did a system generation and copied the distribution zone to initialize the target zone. You could use the ZONERENAME command to rename the copied distribution zone and change the type to target.

### Data Sets Used

The following data sets may be needed to run the ZONERENAME command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see Chapter 34.

SMP_CNTL	SMPLOGA	SMPRPT	<i>newzonename</i>
SMP_CSI	SMPOUT	SMP_SNAP	<i>oldzonename</i>
SMPLOG			

#### Notes:

1. *newzonename* represents the DD statement required for the new zone name. If it is not specified, the data set is dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.
2. *oldzonename* represents the DD statement required for the zone to be renamed by this command. If it is not specified, the data set is dynamically allocated using the ZONEINDEX information in the GLOBALZONE entry.

### Usage Notes

- The *oldzonename* CSI data set need not be mounted when the NEWDATASET operand is used. All operations are performed against the global zone data set and the CSI data set specified as the value of the NEWDATASET operand.
- APPID and ACCID values are not updated in the global zone SYSMOD entries for the renamed zone.
- DDDEF entries in the renamed zone are not changed. To change information in these entries, you must use UCLIN, ZONEEDIT, or the administration dialogs.
- It is your responsibility to ensure that the proper data sets are used for the renamed zone. A partial list to consider is the SMPSCDS, SMPMTS, SMPLOG, SMPSTS, and any target or distribution library data sets.
- If a zone in a copy of a CSI data set is to be renamed, you should be aware that data for other zones in the copied data set remains in the data set, wasting space. Therefore, if you plan to make a copy of a CSI data set and rename the zone therein, you should either use the ZONEDELETE command to delete all the other zones or, when setting up the CSI data sets, ensure that each CSI data set contains only one zone.

### Output

The File Allocation report is produced during ZONERENAME processing. It is described in Chapter 31.



## Examples

The following examples are provided to help you use the ZONERENAME command:

- “Example 1: Renaming an Existing Zone”
- “Example 2: Creating a Target Zone from a Distribution Zone”
- “Example 3: Renaming a Zone in a Copy of a CSI Data Set” on page 438

### Example 1: Renaming an Existing Zone

The simplest use of ZONERENAME is to just assign a new name to an existing zone. For example, assume that responsibility for maintaining the MVS production system has been transferred from department E17 to department C87. The conventions for naming a zone in this establishment is department number plus four characters that help describe the zone's content. You want to change the zone name from E17MVSP to C87MVSP. Assume no changes are required for DDDEF entries in the zone to be renamed, because the target libraries being maintained are not changing.

Before the ZONERENAME operation, the GLOBALZONE ZONEINDEX indicates that zone E17MVSP is in data set SMPE.CSI. There is no ZONEINDEX yet for C87MVSP; it is created by the ZONERENAME command. You now perform the ZONERENAME using the following set of SMP/E commands:

```
SET      BDY(GLOBAL)      /* Set to global.      */
ZONERENAME(E17MVSP)      /* Rename zone E17MVSP */
          TO(C87MVSP)      /* to new name C87MVSP */
          SAMEDATASET      /* within the same CSI. */
                          /* T0TYPE not required */
                          /* because the zone type */
                          /* remains the same.    */
```

After the ZONERENAME operation, the GLOBALZONE ZONEINDEX indicates that zone C87MVSP is on data set SMPE.CSI, and the ZONEINDEX entry for E17MVSP has been removed. The zone definition for E17MVSP in data set SMPE.CSI has been changed so it is now the zone definition for C87MVSP.

### Example 2: Creating a Target Zone from a Distribution Zone

A common use of ZONERENAME involves creating a new target zone after having performed a system generation.

Let us assume you have a distribution zone named MVSDLIB in data set SMPE.CSI. The distribution libraries described by this zone have been used to perform a system generation, and now you want to create a target zone describing the content of your new operating system. You want to call your new target zone MVSPROD. You also want to change the default OPTIONS entry for the renamed zone to MVSOPTS. Further, you want to indicate that the renamed zone is related to the MVSDLIB zone; that is, MVSDLIB is the distribution zone controlling the distribution libraries for the target zone.

First, you make a copy of the SMP.CSI data set containing the distribution zone MVSDLIB (SMPE.CSI.) using the access method services REPRO command. Let us assume the name of the copied data set is PROD.CSI.

Before the ZONERENAME operation, the GLOBALZONE ZONEINDEX indicates that zone MVSDLIB is on data set SMPE.CSI. There is no GLOBALZONE ZONEINDEX for MVSPROD. You can now use the following ZONERENAME command to rename the copy of MVSDLIB in PROD.CSI to MVSPROD, change the zone type to TARGET, and modify the OPTIONS and RELATED subentries:

```

SET      BDY(GLOBAL)      /* Set to global.      */.
ZONERENAME(MVSDLIB)      /* Rename zone MVSDLIB */
      TO(MVSPROD)        /* to new name MVSPROD in a */
      NEWDATASET(        /* copy of a CSI data set */
      PROD.CSI           /* named PROD.CSI.      */
      )                  /*                          */
      TOTYPE(TARGET)      /* Change to a target type. */
      OPTIONS(MVSOPTS)    /* Change default OPTIONS. */
      RELATED(MVSDLIB)    /* Related to DLIB zone.  */.
  
```

After the ZONERENAME operation, the GLOBALZONE ZONEINDEX still indicates that zone MVSDLIB is on data set SMPE.CSI; it has not been removed. A new GLOBALZONE ZONEINDEX entry has been created indicating that zone MVSPROD is on data set PROD.CSI. If the zone definition for MVSPROD is listed, the user sees that the default OPTIONS value is MVSOPTS and that the target zone is related to distribution zone MVSDLIB.

Other than the zone definition, the contents of the renamed zone are unchanged. You must make changes to the DDDEF entries in MVSPROD as required.

After performing these steps, you have a new target zone containing information describing the various modules, macros, and source as they exist on the distribution libraries. The additional information required in the target zone is the description of how the various distribution library elements are combined and installed into the target zone libraries. This information is added to the target zone via the JCLIN command, with stage 1 system generation output as input to SMP/E.

Also, if you are using the dynamic allocation facility in SMP/E, you may have to add or modify some of the DDDEF entries in the new target zone. For an example of priming the new target zone after a full system generation, see "Examples" on page 426.

### Example 3: Renaming a Zone in a Copy of a CSI Data Set

ZONERENAME is also used in making a copy of a target zone or distribution zone. For example, let us suppose you have a target zone named MVSPROD in data set SMPE.CSI, and you want to make a copy of this zone for test purposes. The copy is to be named MVSTEST and is to reside in data set TEST.CSI.

First, you must make a copy of the VSAM CSI data set. This can be done as follows:

- Use access method services (AMS) to define a new CSI data set, to be named TEST.CSI. Do **not** initialize this new CSI with GIMZPOOL; you are going to copy an existing CSI data set into it.
- Use AMS REPRO to copy the original CSI data set, SMPE.CSI, to the newly defined data set, TEST.CSI.

Before the ZONERENAME operation, the GLOBALZONE ZONEINDEX indicates that zone MVSPROD is on data set SMPE.CSI. There is no GLOBALZONE ZONEINDEX for MVSTEST. You now perform the ZONERENAME using the following set of commands:

```
SET      BDY(GLOBAL)      /* Set to global.      */.
ZONERENAME(MVSPROD)      /* Rename zone MVSPROD */.
      TO(MVSTEST)        /* to new name MVSTEST in a */.
      NEWDATASET(        /* copy of a CSI data set */.
          TEST.CSI      /* TEST.CSI.          */.
      )                  /*                      */.
```

After the ZONERENAME operation, the GLOBALZONE ZONEINDEX indicates that zone MVSPROD is on data set SMPE.CSI; it has not been removed. A new GLOBALZONE ZONEINDEX entry has been created, indicating that zone MVSTEST is on data set TEST.CSI.

Other than the zone definition, the content of the renamed zone is unchanged. You must now make changes to the appropriate DDDEF entries in the MVSTEST zone.

---

## Processing

The ZONERENAME command allows you to rename a specified distribution or target zone. SMP/E supports the following variations of renaming:

- Distribution zone to distribution zone
- Target zone to target zone
- Distribution zone to target zone

The rename operation itself is very simple. Before doing it, however, SMP/E makes sure the correct parameters have been entered and that the rename request is valid. If any of the checks fail, an error message is issued, and the rename operation is terminated. The following checks are made:

- *oldzonename* and *newzonename* must not be the same.
- The new zone must not already exist in the data set it is to reside in.
- A GLOBALZONE ZONEINDEX entry for the new zone name must not already exist.
- If **TOTYPE(TARGET)** is specified:
  - The GLOBALZONE ZONEINDEX entry for the old zone name must be a distribution zone.
  - The NEWDATASET operand must also be specified, and the SAMEDATASET operand must **not** be specified.
- If **NEWDATASET** is specified, the data set name specified in the NEWDATASET operand must not be the same as the data set value for the old zone name.
- If **SAMEDATASET** is specified, a GLOBALZONE ZONEINDEX entry must already exist for the old zone name.
- No cross-zone subentry from the zone being renamed refers to the new zone name.

If all validity checking is successful, the renaming can be done. The following operations are done to rename a zone:

- A GLOBALZONE ZONEINDEX entry is added for the new zone name. The data set name is the same as either the value in the old zone (if **SAMEDATASET** was specified) or the name specified in the NEWDATASET operand. The zone type is the same as the old zone type unless the TOTYPE(TARGET) operand is coded, in which case the type is set to TARGET.
- A zone definition entry (either TARGETZONE or DLIBZONE entry) is created for the new zone; the zone definition for the old zone is taken as a base.  
If the RELATED operand or the OPTIONS operand, or both, were specified, that information is used in the zone definition entry; otherwise, that data remains as is in the old zone.
- The old zone definition entry is deleted.
- If the SAMEDATASET operand is specified, the ZONEINDEX entry for the old zone is deleted.

If the zone being renamed contains any TIEDTO subentries for cross-zones, SMP/E issues messages with information to help you update the cross-zones with the new zone name. For information that can help you determine the action to take, see “Usage Notes” on page 393.

---

## Zone and Data Set Sharing Considerations

The following identifies the phases of ZONERENAME processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix E.

### 1. Initialization

Global zone	—	Read without enqueue.
Target zone	—	Read without enqueue.
DLIB zone	—	Read without enqueue.

**Note:** Either the target zone or the distribution zone accessed during this phase is the name specified on the TO operand—that is, the new name of the zone. Except for an error condition, this zone should not exist.

### 2. ZONERENAME processing

Global zone	—	Update with exclusive enqueue.
Target zone	—	Update with exclusive enqueue.
DLIB zone	—	Update with exclusive enqueue.

**Notes:**

- a. Either the target zone or the distribution zone is accessed, according to the zone type specified in the previous SET command.
  - b. The zone accessed in this phase is the zone to be renamed.
- ### 3. Termination
- All resources are freed.

---

## Chapter 31. SMP/E Reports

This chapter describes all the reports produced by SMP/E. The following chart lists these reports and the page on which each is found.

**Note:** The heading of a report indicates the service level of SMP/E that is installed on your system. The SMP/E service level appears as GIMSMP LVL *nn.nn*. For example, GIMSMP LVL 18.1.*nn* is SMP/E Release 8.1 service level *nn*.

Report	Page
CALLLIBS Summary report	442
Causer SYSMOD Summary report	444
CLEANUP Summary report	446
Cross-Zone Requisite SYSMOD report	447
Cross-Zone Summary report	449
Deleted SYSMOD report	453
Element Summary report	455
Exception SYSMOD report	460
File Allocation report	463
GENERATE Summary report	467
JCLIN Cross-Reference report	471
JCLIN Summary report	473
LIST Summary report	477
MOVE/RENAME/DELETE report	479
RECEIVE Exception SYSMOD Data report	485
RECEIVE Summary report	488
REJECT Summary report	492
SOURCEID report	498
SYSMOD Comparison report	500
SYSMOD Regression report	503
SYSMOD Status report	505
UNLOAD Summary report	508
ZONEEDIT Summary report	510
ZONEMERGE report	512

## CALLLIBS Summary Report

This report is produced by the REPORT CALLLIBS command. It provides information about load modules whose LMOD entries contain a CALLLIBS subentry list. The report helps you identify which load modules may need to be relinked after implicitly-included modules have been updated in a library specified in the CALLLIBS subentry list of an LMOD entry.

### Format and Explanation of Data

PAGE <i>nnnn</i> - NOW SET TO GLOBAL ZONE		DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT			
CALLLIBS SUMMARY REPORT FOR <i>ztype</i> ZONE <i>zone</i>					
CALLLIBS DDNAME	LMOD NAME	LMOD SYSLIB	STEPNAME IN JOB <i>zone</i>	CALLLIBS ALLOCATION	DATA SET NAME OR PATH
<i>ddddddd</i>	<i>llllllll</i>	<i>sssssss1</i> <i>sssssss2</i>	<i>LINKxxxx</i> <i>LINKxxxx</i>	<i>ccccccc1</i> <i>ccccccc2</i> <i>cccccccx</i>	<i>nnnnnnnnnnnn...1</i> <i>nnnnnnnnnnnn...2</i> <i>nnnnnnnnnnnn...x</i>

Figure 26. CALLLIBS Summary Report: Standard Format

These are the fields in the report:

***ztype***

indicates the type of zone and can be either a target or DLIB zone.

***zone***

is the name of the zone used as input. The zone name is also used as the job name if punch output is being generated.

**CALLLIBS DDNAME**

is the ddname of the CALLLIBS library being reported on.

**LMOD NAME**

is the name of an LMOD entry containing the CALLLIBS ddname.

**LMOD SYSLIB**

is the ddname of the target library where the load module is installed.

**STEPNAME IN JOB *zone***

is the name of the job step punched by the REPORT CALLLIBS command. This job step contains the JCL to link-edit the load module identified in the LMOD NAME column into the library identified in the LMOD SYSLIB column.

**Notes:**

1. This column is blank and the column heading will not contain a job name if JCL is not being punched.
2. The job steps are named LINKxxxx where xxxx is the step (0001 through 9999) within that job.

**CALLLIBS ALLOCATION**

is the list of DDDEF entries that make up the LMOD entry's CALLLIBS subentry list.

**DATA SET NAME OR PATH**

is the name of the data set or path in the DDDEF entry listed in the CALLLIBS ALLOCATION column. The full data set name (up to 44 characters) and full pathname (up to 255 characters) are shown.

**Note:** If SMPPUNCH output was produced for the REPORT CALLLIBS command, the generated JCL includes the appropriate information from the DDDEF entry to allocate this data set or path.

The pathname is enclosed by apostrophes. If the pathname with its enclosing apostrophes is greater than 63 characters, it will run to the next line in the report.

If no DDDEF entry was found for the CALLLIBS ddname in the zone being reported on, you will see **NO DDDEF** in the DATA SET NAME OR PATH column.

**Example: CALLLIBS Summary Report for a Target Zone**

PAGE <i>nnnn</i> - NOW SET TO GLOBAL ZONE						DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT
CALLLIBS SUMMARY REPORT FOR TARGET ZONE MVSTZ1						
CALLLIBS DDNAME	L MOD NAME	L MOD SYSLIB	STEPNAME IN JOB MVSTZ1	CALLLIBS ALLOCATION	DATA SET NAME OR PATH	
CSSLIB	IEELOAD1	LINKLIB	LINK0001	CSSLIB	SYS1.CSSLIB	
	IEFLOAD5	LINKLIB	LINK0001	CSSLIB	SYS1.CSSLIB	
	IGGLOAD3	LPALIB	LINK0002	CSSLIB PLIBASE	SYS1.CSSLIB SYS1.PLIBASE	
	IWWLOAD6	LPALIB	LINK0002	CSSLIB PLIBASE	SYS1.CSSLIB SYS1.PLIBASE	
HFSCLIB1	BPXLMOD4	BPXLIB1	LINK0003	HFSCLIB1	'/this/pathname/fits/on/one/record/in/the/report/'	
HFSCLIB2	BPXLMOD5	BPXLIB1	LINK0004	HFSCLIB2	'/this/is/a/very/long/pathname/it/requires/more/than/a/single/r ecord/to/display./it/also/contains/'/a/single/quote/'	
PLIBASE	IGGLOAD3	LPALIB	LINK0002	CSSLIB PLIBASE	SYS1.CSSLIB SYS1.PLIBASE	
	IWWLOAD6	LPALIB	LINK0002	CSSLIB PLIBASE	SYS1.CSSLIB SYS1.PLIBASE	

Figure 27. CALLLIBS Summary Report: Sample Report

## Causer SYSMOD Summary Report

To reduce the work needed to determine which errors caused SYSMODs to fail, SMP/E performs root cause analysis for the ACCEPT, APPLY, and RESTORE commands to identify causer SYSMODs. A causer SYSMOD is a SYSMOD whose failure has led to the failure of other SYSMODs. The types of errors SMP/E analyzes to determine causer SYSMODs include the following:

- Held SYSMODs
- Missing requisite SYSMODs
- Utility program failures: copy, update, assembler, link-edit utility, zap
- Out-of-space conditions: x37 ABENDs
- Missing DD statements and other allocation errors
- ID errors (a SYSMOD does not supersede or specify as a prerequisite an RMID or a UMID of an element)
- JCLIN errors (syntax errors)

The Causer SYSMOD Summary report lists the causer SYSMODs and a summary of the related messages describing the errors that need to be fixed to successfully process the SYSMODs. (Subsequent messages generated because of these initial errors are not included in the report.)

This report is produced during the processing of APPLY, ACCEPT, and RESTORE commands. It is **not** produced when there are no errors or when there are certain types of errors, such as unusual VSAM errors (as indicated by GIM443xx messages).

## Format and Explanation of Data

```
PAGE nnnn - NOW SET TO TARGET ZONE nnnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
CAUSER SYSMOD SUMMARY REPORT FOR xxxxxxxx PROCESSING
CAUSER      FMID      MESSAGE ID  PAGE      ERROR DESCRIPTION AND POSSIBLE CAUSES
aaaaaaa    bbbbbbb    cccccccc   dd        eeeeeee....
aaaaaaa    bbbbbbb    cccccccc   dd        eeeeeee....
aaaaaaa    bbbbbbb    cccccccc   dd        eeeeeee....
aaaaaaa    bbbbbbb    cccccccc   dd        eeeeeee....
aaaaaaa    bbbbbbb    cccccccc   dd        eeeeeee....
```

Figure 28. Causer SYSMOD Summary Report: Standard Format

These are the fields in the report:

xxxxxxx

is the SMP/E command being processed: APPLY, ACCEPT, or RESTORE.

### CAUSER

identifies the SYSMOD whose failure led to the failure of other SYSMODs.

### FMID

identifies the FMID for the causer SYSMOD when that information is available. Otherwise, the field is blank.



**MESSAGE ID**

is the message identification number for the message describing the error that caused this SYSMOD to fail (8-character alphanumeric string plus a 1-character severity value).

**PAGE**

identifies the page number in SMPOUT output where the SYSMOD failure message is located.

**ERROR DESCRIPTION AND POSSIBLE CAUSES**

provides a summary of the error and, when feasible, a list of possible causes of the error.

**Example: Causer SYSMOD Summary Report for APPLY Processing**

```

PAGE nnnn - NOW SET TO TARGET ZONE nnnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
CAUSER SYSMOD SUMMARY REPORT FOR APPLY PROCESSING
CAUSER   FMID   MESSAGE ID  PAGE   ERROR DESCRIPTION AND POSSIBLE CAUSES
AZ00001           GIM37301E    1   THIS SYSMOD HAS MORE THAN ONE APPLICABLE ++VER.
EYC0437   HAE1500  GIM24001I   30   ASSEMBLER ERROR FOR MODULE IATDMER IN LIBRARY LINKLIB FOR SYSMOD
        EYC0437. THE SEQUENCE NUMBER IS 00001.
        --- POSSIBLE CAUSES ---
        1. THE CORRECT DEFAULT UTILITY RETURN CODE WAS NOT SPECIFIED.
        2. THE OPTIONS ENTRY DOES NOT CONTAIN THE CORRECT UTILITY ENTRY.
UP42613   HAE1500  GIM35902I    6   SYSTEM HOLD DEP WAS NOT RESOLVED.
UY12605   HAE1500  GIM40501E   30   THE DISTLIB IN ++MOD BLMGDBWR IS DIFFERENT FROM THE ELEMENT ENTRY IN THE ZONE.
        GIM40501E   30   THE DISTLIB IN ++MOD BLMGDBWI IS DIFFERENT FROM THE ELEMENT ENTRY IN THE ZONE.
UZ62368   JTC2412  GIM35901I    2   ERROR HOLD AZ71745 IS NOT RESOLVED.
        GIM35901I    2   ERROR HOLD AZ75533 IS NOT RESOLVED.
UZ75098   HJE2330  GIM35909I   27   COREQUISITE SYSMOD UZ75006 WAS EXCLUDED.
    
```

Figure 29. Causer SYSMOD Summary Report: Sample Report for APPLY

## CLEANUP Summary Report

This report is produced during CLEANUP processing to summarize the elements or aliases deleted from each target zone work data set. If no entries were deleted (for example, there were no entries or an abend occurred before report data was collected), the CLEANUP report states THERE WAS NO CLEANUP DONE.

### Format and Explanation of Data

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

      CLEANUP SUMMARY REPORT

NOTE: * - INDICATES THE ENTRY IS AN ALIAS OF THE PRECEDING MAJOR MEMBER NAME

DDNAME  ENTRY DELETED

aaaaaaa bbbbbbbb bbbbbbbb bbbbbbbb bbbbbbbb bbbbbbbb
aaaaaaa bbbbbbbb bbbbbbbb bbbbbbbb bbbbbbbb bbbbbbbb
aaaaaaa bbbbbbbb bbbbbbbb bbbbbbbb bbbbbbbb bbbbbbbb
aaaaaaa bbbbbbbb bbbbbbbb bbbbbbbb bbbbbbbb bbbbbbbb
```

Figure 30. CLEANUP Summary Report: Standard Format

These are the fields in the report:

**DDNAME**

is the ddname of the data set from which data was deleted: SMPSCDS, SMPMTS, or SMPSTS. Each ddname is shown only once.

**ENTRY DELETED**

is a list of the elements or alias names deleted from the data set. If there is an alias name, it follows the real name of the member and is preceded by an asterisk.

### Example: CLEANUP Summary Report

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

      CLEANUP SUMMARY REPORT

NOTE: * - INDICATES THE ENTRY IS AN ALIAS OF THE PRECEDING MAJOR MEMBER NAME

DDNAME  ENTRY DELETED

SMPSCDS UZ00010  UZ00020
SMPMTS  MAC02   MAC03
SMPSTS  SRC11   SRC12
```

Figure 31. CLEANUP Summary Report: Sample Report

## Cross-Zone Requisite SYSMOD Report

This report is produced at the completion of REPORT CROSSZONE processing when cross-zone requisite checking was done and requisites in installed SYSMODs affected functions in the selected zones. The report shows the affected zones, the requisites that must be installed in each zone, and the SYSMODs that contained ++IF statements naming the requisites. The requisite SYSMODs are listed alpha-numerically by SYSMOD ID for each FMID in each zone.

## Format and Explanation of Data

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

      CROSSZONE REQUISITE SYSMOD REPORT FOR xxxxxx

  _ZONE_   _REQUIRES_   _CAUSER_
  NAME     FMID        SYSMOD   RECEIVED SYSMOD   FMID     ZONE
  zzzzzzz ffffffff sssssss xxx      ttttttt rrrrrrr wwwwwwwww
          ffffffff sssssss xxx      ttttttt rrrrrrr wwwwwwwww
  zzzzzzz ffffffff sssssss xxx      ttttttt rrrrrrr wwwwwwwww
          ffffffff sssssss xxx      ttttttt rrrrrrr wwwwwwwww
    
```

Figure 32. Cross-Zone Requisite SYSMOD Report: Standard Format

These are the fields in the report:

**xxxxxxx**

is the command to be used to install the SYSMODs in the report: ACCEPT if the report was done for distribution zones, or APPLY if the report was done for target zones.

**ZONE NAME**

is the zone where requisite SYSMODs must be installed.

**REQUIRES FMID**

is the FMID for which requisite SYSMODs must be installed.

**REQUIRES SYSMOD**

is the ID of a requisite SYSMOD. If there are no requisite SYSMODs for a particular zone, you see NONE in the REQUIRES SYSMOD field.

**RECEIVED**

indicates whether the requisite SYSMOD has been received. YES in the RECEIVED field means the SYSMOD has been received, and NO means it has not. You must receive all requisite SYSMODs before installing them.

**CAUSER SYSMOD**

is the ID of the SYSMOD that contained the ++IF statement for the requisite.

**CAUSER FMID**

is the FMID for the causer SYSMOD. If UNKNOWN is in the CAUSER FMID field, the causer SYSMOD was installed with a release of SMP/E before Release 3. This is not an error.

**CAUSER ZONE**

is the zone that contained the causer SYSMOD.

**Example: Cross-Zone Requisite SYSMOD Report**

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
```

CROSSZONE REQUISITE SYSMOD REPORT FOR APPLY

ZONE NAME	FMID	REQUIRES SYSMOD	RECEIVED	SYSMOD	CAUSER FMID	ZONE
MVS370		NONE				
PRODESA	HJS1201	UZ00027	YES	UZ00025	EJS1201	PROD370
	HJS1201	UZ00028	NO	UZ00026	EJS1201	PROD370
PROD370	EJS1201	UZ00023	YES	UZ00011	EBB1202	MVS370
	EJS1201	UZ00024	YES	UZ00013	EBB1202	MVS370

Figure 33. Cross-Zone Requisite SYSMOD Report: Sample Report

## Cross-Zone Summary Report

This report, produced during APPLY and RESTORE processing, summarizes the cross-zone work that has been done, the cross-zone work that has not been done, and why. The cross-zone work resulting from renamed LMODs is summarized in the Move/Rename/Delete report.

### Format and Explanation of Data

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

                CROSS-ZONE SUMMARY REPORT FOR xxxxxx yyyyyy PROCESSING

NOTE: SEE THE MOVE/RENAME/DELETE REPORT FOR A SUMMARY OF CROSS-ZONE WORK CAUSED BY RENAMED LMODS

CROSS          LMOD          LKED
ZONE   LMOD   SYSLIB  MODULE  ACTION  RC  REASON  SYSMOD

aaaaaaa bbbbbbbb ccccccc dddddd eeeeeeeeeeeeeeeeeeeee fff ggggggggggggggggggggggggggggg hhhhhh
    
```

Figure 34. Cross-Zone Summary Report

These are the fields in the report:

**xxxxxxx**

is the SMP/E command being processed: APPLY or RESTORE.

**yyyyyyy**

is CHECK if CHECK was specified on the command. Otherwise, this field is blank.

**CROSS-ZONE**

is the cross-zone name.

**LMOD**

is the name of the cross-zone LMOD being updated.

**LMOD SYSLIB**

is the SYSLIB for the cross-zone LMOD being updated.

**Note:** SMPLTS appears in this column when a module is linked into a base version of a load module in the SMPLTS data set.

**MODULE**

is the name of the changed module from the set-to zone that caused the cross-zone LMOD to be updated.

**ACTION**

indicates whether the cross-zone work was done.

**MOD LINKED INTO LMOD**

The module from the set-to zone was replaced in the cross-zone LMOD.

**MOD NOT LINKED INTO LMOD**

The module from the set-to zone was not replaced in the cross-zone LMOD.

**Note:** If the SYSLIB value is not filled in and the load module exists in two SYSLIBs, the module from the set-to zone may have been successfully link-edited to the load module's first SYSLIB. Use the output from the link-edit utility to determine whether the link-edit was successful for the load module's first SYSLIB.

### **ZAP LINKED INTO LMOD**

The zapped module from the set-to zone has been replaced in the cross-zone LMOD.

### **ZAP NOT LINKED INTO LMOD**

The zapped module from the set-to zone has not been replaced in the cross-zone LMOD.

**Note:** If the SYSLIB value is not filled in and the load module exists in two SYSLIBs, the module from the set-to zone may have been successfully link-edited to the load module's first SYSLIB. Use the output from the link-edit utility to determine whether the link-edit was successful for the load module's first SYSLIB.

### **MOD DELETED FROM LMOD**

The module has been successfully deleted from the cross-zone LMOD.

### **MOD NOT DELETED FROM LMOD**

The module has not been deleted from the cross-zone LMOD.

**Note:** If the SYSLIB value is not filled in and the load module exists in two SYSLIBs, the module from the set-to zone may have been successfully link-edited to the load module's first SYSLIB. Use the output from the link-edit utility to determine whether the link-edit was successful for the load module's first SYSLIB.

### **LKED RC**

is the return code from the link-edit utility.

### **REASON**

is the reason the cross-zone work was not done.

**ABEND** The cross-zone work could not be done because of an abend. You should use a combination of the LINK command, the UCLIN command, or the link-edit utility outside of SMP/E to complete the cross-zone work.

**Note:** If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

### **SEVERE ERROR ENCOUNTERED**

The cross-zone work could not be done, because a severe error was encountered. You should use a combination of the LINK command, the UCLIN command, or the link-edit utility outside of SMP/E to complete the cross-zone work.

**Note:** If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

**CROSS-ZONE PROCESSING DEFERRED**

The cross-zone work could not be done, because the cross-zone XZLINK subentry was set to DEFER. You should use a combination of the LINK command, the UCLIN command, or the link-edit utility outside of SMP/E to complete the cross-zone work.

**Note:** If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

**ERROR FOUND FOR ZONE**

The cross-zone work could not be done, because of an error found while processing the cross-zone. You should use a combination of the LINK command, the UCLIN command, or the link-edit utility outside of SMP/E to complete the cross-zone work.

**Note:** If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

**CSI DATA SET UNAVAILABLE**

The cross-zone work could not be done, because the CSI data set containing the cross-zone was not available. You should use a combination of the LINK command, the UCLIN command, or the link-edit utility outside of SMP/E to complete the cross-zone work.

**Note:** If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

**LMOD NOT IN ZONE**

The cross-zone work for the LMOD was not done, because the LMOD no longer exists in the cross-zone. No action is required to complete the cross-zone work.

**LMOD DOES NOT REFER TO MOD**

The module from the set-to zone has an XZLMOD subentry, which means it was once part of the cross-zone LMOD. However, the cross-zone LMOD was not updated to include the updated module, because the LMOD does not have an XZMOD subentry for the module. If the cross-zone LMOD no longer needs the MOD, no action is required. Otherwise, use the LINK command to link the module into the cross-zone LMOD.

**LMOD NOT IN SYSLIB *ddname***

The cross-zone work for the LMOD could not be done, because the LMOD does not exist in its SYSLIB. If the SYSLIB is incorrect, use UCLIN to correct it; then use the LINK command or the link-edit utility outside of SMP/E to complete the cross-zone work for the LMOD.

**CANNOT ALLOCATE SYSLIB *ddname***

The cross-zone work for the LMOD could not be done, because its SYSLIB could not be allocated. Fix the allocation error, and then

use the LINK command or the link-edit utility outside of SMP/E to complete the cross-zone work for the LMOD.

**USABLE COPY OF ZAP NOT FOUND**

The zap could not be included in the cross-zone LMOD, because a single-CSECT LMOD containing just the zapped module could not be found. Find a usable copy of the module with the zap, and link it into the cross-zone LMOD using either the LINK command or the link-edit utility outside of SMP/E. If you cannot find a usable copy, use the ZAP utility to update the cross-zone LMOD.

**MODULE NO LONGER BEING PROCESSED**

The cross-zone work was not done, because the module from the set-to zone is no longer selected to be updated. No action is required to complete the cross-zone work. It is automatically done once the module from the set-to zone has been successfully updated by a subsequent APPLY or RESTORE command.

**ASSEMBLY NOT AVAILABLE**

The assembled module was not available, because the assembly was not done. SMP/E determined that no LMODs from the set-to zone needed the assembled module and, therefore, did not do the assembly work. To determine the action you need to take, see the *Programmer Response* of message GIM69136.

**SYSMOD**

is ID of the SYSMOD causing the cross-zone work to be scheduled.

**Example: Cross-Zone Summary Report for APPLY Processing**

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
    
```

CROSS-ZONE SUMMARY REPORT FOR APPLY PROCESSING

CROSS ZONE	LMOD	LMOD SYSLIB	MODULE	ACTION	LKED RC	REASON	SYSMOD
XZONE5	XLMOD1	LOADLIBR	MOD110	MOD DELETED FROM LMOD	00	N/A	APP1003
ZONE10	LMOD8	N/A	MOD110	MOD NOT DELETED FROM LMOD	N/A	ERROR FOUND FOR ZONE	APP1003
ZONE9	LMOD7	N/A	MOD110	MOD NOT DELETED FROM LMOD	N/A	ERROR FOUND FOR ZONE	APP1003

Figure 35. Cross-Zone Summary Report: Sample Report for APPLY



## Deleted SYSMOD Report

This report is produced at the completion of APPLY and ACCEPT processing when SMP/E has processed a SYSMOD with a ++VER DELETE statement. The report shows the function SYSMODs that have been deleted, and all the service SYSMODs that were applicable to those functions.

### Format and Explanation of Data

PAGE <i>nnnn</i> - NOW SET TO <i>zzzzz</i> ZONE <i>nnnnnn</i> DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT					
DELETED FUNCTION REPORT FOR <i>xxxxxxx</i> <i>yyyy</i> PROCESSING					
SYSMOD CAUSING THE DELETION	DELETED THE FOLLOWING TYPE	SYSMODS SYSMOD			
<i>aaaaaaa</i>	<i>bbbbbbb</i>	<i>ccccccc ccccccc ccccccc</i>			
	<i>bbbbbbb</i>	<i>ccccccc ccccccc ccccccc</i>			
<i>aaaaaaa</i>	<i>bbbbbbb</i>	<i>ccccccc</i>			

Figure 36. Deleted SYSMOD Report: Standard Format

These are the fields in the report:

*xxxxxxx*

is the SMP/E command being processed: APPLY or ACCEPT.

*yyyy*

is CHECK if the CHECK operand was specified on the APPLY or ACCEPT command. Otherwise, this field is blank.

#### SYSMOD CAUSING THE DELETION

identifies the SYSMOD containing the ++VER DELETE statement.

#### DELETED THE FOLLOWING SYSMODS

identifies the type and SYSMOD ID of all the SYSMODs that were deleted. The type may be APAR, FUNCTION, PTF, or USERMOD. The SYSMOD IDs are listed from left to right next to the type. Each PTF, APAR, and USERMOD listed in the TYPE field belongs to the function SYSMOD listed immediately above it.

When the type is FUNCTION, the value of the SYSMOD can be one of the following:

##### SYSMOD ID only

The SYSMOD has been installed on the target or distribution libraries and is specified in the DELETE operand list of the ++VER statement for the deleting SYSMOD.

##### SYSMOD ID followed by FMID(*sysmod-id*)

The SYSMOD is a dependent function that has been implicitly deleted. FMID identifies the associated base function that has been explicitly deleted.

## SYSMOD ID followed by NOT PREVIOUSLY INSTALLED

The SYSMOD has been specified in the DELETE operand list of the ++VER statement for the deleting SYSMOD, but has not been installed on your target or distribution libraries.

## SYSMOD ID followed by PREVIOUSLY DELETED

The SYSMOD has been specified in the DELETE operand list of the ++VER statement for the deleting SYSMOD, but has been previously deleted by another function SYSMOD.

SYSMODs that have been previously deleted may remain as entries on the target zone or distribution zone if they are specified in the SUP operand list either of the deleting function SYSMOD or of another SYSMOD that has been processed concurrently.

## Example: Deleted SYSMOD Report for APPLY

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

DELETED FUNCTION REPORT FOR APPLY PROCESSING

SYSMOD      DELETED THE
CAUSING THE FOLLOWING SYSMODS
DELETION     TYPE      SYSMOD
-----
FYZ3000     FUNCTION  FYZ1000
            PTF      UZ00111  UZ00123  UZ00135
            USERMOD  MY11111

            FUNCTION  GYZ1010  FMID (FYZ1000)
            PTF      UZ00112  UZ00124  UZ00136
            USERMOD  MY11112

            FUNCTION  GYZ1020  FMID (GYZ1010)
            PTF      UZ00142  UZ00164
            APAR      AZ12345

            FUNCTION  FYZ2000  NOT PREVIOUSLY INSTALLED
```

Figure 37. Deleted SYSMOD Report: Sample Report for APPLY

## Element Summary Report

This report is produced at the completion of APPLY, ACCEPT, and RESTORE processing to describe the status of the libraries that were updated for each macro, source, module, or data element. Elements are grouped by element type, in the order indicated below, and are listed alphabetically under each element type:

1. Macros (++MAC and ++MACUPD statements)
2. Source (++SRC and ++SRCUPD statements)
3. Modules (++MOD and ++ZAP statements)
4. Data elements (++*element* statements)
5. HFS elements (++HFS statements)

The report is not generated when processing for all selected SYSMODs stops before any elements are selected.

## Format and Explanation of Data

PAGE <i>nnnn</i> - NOW SET TO <i>zzzzz</i> ZONE <i>nnnnnn</i> DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT											
ELEMENT SUMMARY REPORT FOR <i>xxxxxxx</i> <i>yyyyy</i> PROCESSING											
ELEMENT TYPE	ELEMENT NAME	ELEMENT STATUS	CURRENT FMID	CURRENT RMID	DISTLIB LIBRARY	SYSLIB LIBRARY	ASSEM NAMES	LOAD MODULE	LMOD SYSLIB	SYSMOD NAME	SYSMOD STATUS
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>ccccccc</i>	<i>dddddd</i>	<i>eeeeee</i>	<i>ffffff</i>	<i>gggggg</i>	<i>hhhhhhh</i>	<i>iiiiiii</i>	<i>jjjj</i>	<i>kkkkkk</i>	<i>lllllll</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>ccccccc</i>	<i>dddddd</i>	<i>eeeeee</i>	<i>ffffff</i>	<i>gggggg</i>	<i>hhhhhhh</i>	<i>iiiiiii</i>	<i>jjjj</i>	<i>kkkkkk</i>	<i>lllllll</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>ccccccc</i>	<i>dddddd</i>	<i>eeeeee</i>	<i>ffffff</i>	<i>gggggg</i>	<i>hhhhhhh</i>	<i>iiiiiii</i>	<i>jjjj</i>	<i>kkkkkk</i>	<i>lllllll</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>ccccccc</i>	<i>dddddd</i>	<i>eeeeee</i>	<i>ffffff</i>	<i>gggggg</i>	<i>hhhhhhh</i>	<i>iiiiiii</i>	<i>jjjj</i>	<i>kkkkkk</i>	<i>lllllll</i>

Figure 38. Element Summary Report: Standard Format

These are the fields in the report:

*xxxxxxx*

is the SMP/E command being processed: APPLY, ACCEPT, or RESTORE.

*yyyyy*

is CHECK if the CHECK operand was specified on the command. Otherwise, this field is blank.

### ELEMENT TYPE

is the element type: MAC, MOD, SRC, MUPD, SUPD, S/ZAP, HFS, or one of the data element types shown in Table 26 on page 523.

### ELEMENT NAME

is the element name.

### ELEMENT STATUS

describes what happened to the element. It may be one of the following:

#### APPLIED, ACCEPTED, or RESTORED

The element was processed successfully.

**BYPASS** SMP/E detected an error while checking MODIDs. However, because **BYPASS(ID)** was specified, the element was processed.

**DELETED**

The element was selected and deleted. A element may be deleted if:

- **DELETE** was specified on the element MCS.
- The supplying SYSMOD was deleted. (A SYSMOD that was applied or accepted specified the supplying SYSMOD on the ++VER DELETE statement.)
- The supplying SYSMOD was restored.

**Note:** If the SYSMOD being restored added an existing module to an existing load module, RESTORE processing does not remove that module from the target libraries or from the load module.

Typically, in such cases, the SYSMOD added new modules that called the existing module in the load module. So, when the SYSMOD is restored, those new modules are deleted from the target libraries, and the load module is relinked to remove the new modules. As a result, although the existing module is still physically in the load module, it has been logically removed, because the new modules that called it are gone.

**DLIB ERR**

The DISTLIB value on the MCS does not match the DISTLIB value in the target or distribution zone element entry. The element was not processed, and the SYSMOD status is NOGO.

**INCMPLT** Element processing is incomplete because of some failure. No libraries were updated.

**ID ERR** SMP/E detected an error while checking MODIDs. The element was not processed. See the messages in SMPOUT to determine the error.

**NOGO** The element was not processed if the SYSMOD status is also NOGO. If the SYSMOD status is ERROR, the element may have been processed. Check the messages in SMPOUT for the status of the library in which the element resides.

**NOT SEL** This version of the element was not selected. Following are the reasons an element might not be selected.

- **Multiple versions of the element.** If multiple versions of the same element are being processed concurrently, a superior version may have been chosen from another SYSMOD.

A module might not be selected if a macro or source caused a higher level of the module to be assembled.

If none of the versions of an element are selected, a superior version already exists on the target system.

- **FMID mismatch.** Often, when an element is not selected, its FMID did not match the FMID of the element on the target system. The selection and exclusion of elements is discussed in "Processing" on pages 28 and 77.

- **No target library.** When an element has no target library (as described in message GIM43401), the status is NOT SEL. The element is not selected for update to any target libraries.
- **Owning SYSMOD marked NOGO.** An element can be marked NOT SEL if the owning SYSMOD is marked NOGO before processing is finished for that element.

**Note:** NOT SEL refers to processing in the set-to zone. An element with a status of NOT SEL can be updated in cross-zone LMODs.

**SRC SEL** Because the source version of the module was selected, the object version (++MOD) was not processed. For example, when a source or macro is changed, the source may be reassembled to create the updated object module.

**CURRENT FMID**

is the FMID that appears in the element entry at the end of command processing. It appears only if the element is processed successfully.

**CURRENT RMID**

is the RMID that appears in the element entry at the end of command processing. It appears only if the element is processed successfully.

**DISTLIB LIBRARY**

is the ddname of the distribution library containing the element.

**SYSLIB LIBRARY**

is the ddname of the target library containing the element.

**ASSEM NAMES**

is a list of SRC or ASSEM modules assembled as a result of a macro or source modification. This field is not present for ACCEPT processing. It is present for RESTORE processing only if the MAC entry contains a GENASM subentry.

**LOAD MODULE**

is a list of load modules that were link-edited or copied using the module in the ELEMENT NAME field. This field is not present for ACCEPT processing.

**Note:** For S/ZAP elements, this list shows the load modules that included the module specified on the ++ZAP MCS.

If two operands were used on the IMASPZAP NAME statement to limit the load modules that were updated, this list shows all the load modules that the element is part of. Check the utility completion messages (GIM237xx) to determine which load modules were updated.

**LMOD SYSLIB**

is a list of target libraries that contained the load module in the LOAD MODULE field and that were updated during APPLY or RESTORE processing. This field is set to the DISTLIB value for ACCEPT processing.

**Note:** SMPLTS appears in this column when a module is linked into a base version of a load module in the SMPLTS data set.

**SYSMOD NAME**

identifies the SYSMODs that changed the element in the ELEMENT NAME field.

### **SYSMOD STATUS**

is the status of the SYSMOD in the SYSMOD NAME field. It may be one of the following:

#### **APPLIED, ACCEPTED, or RESTORED**

The SYSMOD was successfully processed.

#### **DELETED**

The SYSMOD was explicitly or implicitly deleted.

**ERROR** SYSMOD processing stopped after some target libraries or SMP/E libraries were updated, but before the SYSMOD was completely processed. A SYSMOD is completely processed when all its elements have been processed and all of the SYSMOD's requisites have been completely processed. See SMPDOUT to determine the cause of the error.

**Note:** ERROR does not appear in the SYSMOD status field when the CHECK operand is specified on the command.

#### **EXCLUDED**

The SYSMOD was specified on the EXCLUDE operand.

**HELD** The SYSMOD was held because one or more of HOLD reason IDs were not resolved.

**INCMPLT** SYSMOD processing is incomplete because of some failure. No target libraries were updated.

**NOGO** The SYSMOD was not processed before any updates. This can happen when a related SYSMOD has an error. See SMPDOUT to determine the cause of the error.

**NOGO(E)** SYSMOD processing stopped because a required SYSMOD was excluded.

**NOGO(H)** SYSMOD processing stopped because a required SYSMOD was held.

**SUPD** The SYSMOD is superseded by one or more SYSMODs being processed. The superseding SYSMODs are shown in the REQUISITE AND SUPBY SYSMODS field in the SYSMOD status report.

**Example: APPLY CHECK Element Summary Report**

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
ELEMENT SUMMARY REPORT FOR APPLY CHECK PROCESSING
ELEMENT  ELEMENT  ELEMENT  CURRENT  CURRENT  DISTLIB  SYSLIB  ASSEM  LOAD  LMOD  SYSMOD  SYSMOD
TYPE      NAME      STATUS   FMID     RMID     LIBRARY  LIBRARY  NAMES  MODULE  SYSLIB  NAME     STATUS
MAC       MAC1     APPLIED  F000000  F000000          MACLIB                F000000  APPLIED
SRC       SRC1     APPLIED  F000000  F000000          SRCLIB                F000000  APPLIED
MOD       MOD1     APPLIED  F000000  P000001          MOD1  LINKLIB  P000001  APPLIED
          NOT SEL  F000000  APPLIED  P000002          MOD2  LINKLIB  P000002  APPLIED
MOD       MOD2     APPLIED  F000000  P000002          MOD2  LINKLIB  P000002  APPLIED
          NOT SEL  F000000  APPLIED
:
CLIST     ZCLIST   APPLIED  F000000  F000000          SXYZLIST              F000000  APPLIED
PARM      BPARM    APPLIED  F000000  F000000          SXYZPARM              F000000  APPLIED
HFS       HFSEL1   APPLIED  HFSFUNC  HFSPTF1  APOSIXL1 HFSTGT1              HFSPTF1  APPLIED
HFS       HFSEL2   APPLIED  HFSFUNC  HFSPTF2  APOSIXL2 HFSTGT2              HFSPTF2  APPLIED
    
```

Figure 39. Element Summary Report: Sample Report for APPLY CHECK

## Exception SYSMOD Report

This report is produced at the completion of REPORT ERRSYSMODS processing during which exception SYSMOD checking was done and HOLDERROR reason IDs were not resolved for SYSMODs installed in the specified zone. The report shows the exception SYSMODs that were previously installed, the HOLDERROR reason IDs that have made them exception SYSMODs, resolving SYSMODs that have been received but not yet installed, and the ++HOLD statements for the HOLDERROR reason IDs.

The exception SYSMOD reports produced by a given REPORT ERRSYSMODS command are arranged alphanumerically by zone name. Each report begins on a new page. Within each report, the exception SYSMODs are listed alphanumerically by SYSMOD ID.

## Format and Explanation of Data

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
EXCEPTION SYSMOD REPORT FOR ZONE xxxxxxx DATE: vv/vv/vv - ww/ww/ww

NOTE: * - INDICATES THAT THE RESOLVING SYSMOD IS HELD. TO FIND THE RESOLVING
      SYSMODS FOR HELD SYSMODS,
      RUN REPORT ERROR SYSMODS, SPECIFYING THE GLOBAL ZONE

SYSMOD  HOLD   REASON  RESOLVING  ++HOLD  ERROR DATA
NAME     FMID   IDS     SYSMODS

aaaaaaa bbbbbb  ccccc  ddddd    eeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
          ddddd    eeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
          ddddd    eeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
          ccccc  ddddd    eeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
          eeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
          eeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

aaaaaaa bbbbbb  ccccc  ddddd    eeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
          ddddd    eeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
          ddddd    eeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
  
```

Figure 40. Exception SYSMOD Report: Standard Format

These are the fields in the report:

**xxxxxx**

is the name of the zone being reported on. A separate report is provided for each zone specified on the ZONES operand of the REPORT ERRSYSMODS command.

**DATE:** vv/vv/vv - ww/ww/ww

shows the beginning and ending dates of the HOLDDATA used to create this report. The dates appear as mm/dd/yy, where mm is the month (01–12), dd is the day (01–31), and yy is the year (00–99).

- vv/vv/vv is the beginning date of the HOLDDATA used to create this report. If the BEGINDATE operand was specified on the REPORT ERRSYSMODS command, that value appears in this field. Otherwise, the word THROUGH appears.
- ww/ww/ww is the ending date of the HOLDDATA used to create this report.



If the ENDDATE operand was specified on the REPORT ERRSYSMODS command, that value appears in this field. Otherwise, the current date or the IPL date (if any) specified on the EXEC statement for GIMSMP is used.

If neither **BEGINDATE** nor **ENDDATE** was specified on the REPORT ERRSYSMODS command, no dates appear in this field.

**SYSMOD NAME**

is the ID of a SYSMOD installed in the specified zone that meets these conditions:

- For target and distribution zones, it is an installed SYSMOD for which ++HOLD statements were received later and whose ERROR reason IDs have not yet been resolved.
- For the global zone, it is a received SYSMOD for which ++HOLD statements with ERROR reason IDs have been received.

If there are no SYSMODs to report on in the zone, you see \*\*\*NONE in the SYSMOD NAME field, and the remaining fields are blank.

**HOLD FMID**

is the FMID of the ++HOLD MCS that was received for the HOLDERERROR reason ID.

**REASON IDS**

is a list of one or more HOLDERERROR reason IDs (APAR numbers) that caused the installed SYSMOD to become an exception SYSMOD.

**RESOLVING SYSMODS**

is a list of resolving SYSMODs received but not yet installed. A resolving SYSMOD may either match the HOLDERERROR reason ID or may specifically supersede it.

- For a target zone, these SYSMODs must be received but not applied.
- For a distribution zone, these SYSMODs must be received but not accepted.
- For the global zone, these SYSMODs must be received.

If no resolving SYSMODs have been received for an exception SYSMOD, you see \*\*\*NONE in the RESOLVING SYSMODS field.

If a resolving SYSMOD is itself held for an error reason ID, you see \* after the SYSMOD ID. You can run REPORT ERRSYSMODS, specifying the global zone, to see whether any SYSMODs have been received that resolve this additional hold.

**++HOLD ERROR DATA**

is the ++HOLD MCS that was received for the HOLDERERROR reason ID. This data may span several lines and is provided for each error reason ID.

Example: Exception SYSMOD Report

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
EXCEPTION SYSMOD REPORT FOR ZONE DZONE1 DATE: 07/01/90 - 08/01/90

NOTE: * - INDICATES THAT THE RESOLVING SYSMOD IS HELD. TO FIND THE RESOLVING
      SYSMODS FOR HELD SYSMODS,
      RUN REPORT ERROR SYSMODS, SPECIFYING THE GLOBAL ZONE

I  SYSMOD  HOLD   REASON  RESOLVING  ++HOLD  ERROR DATA
   NAME    FMID   IDS     SYSMODS

I  UZ00001  HBB1200  AZ65432  ***NONE   ++HOLD (UZ00001) FMID(HBB1200)
I                                     ERROR REASON(AZ65432)
I                                     COMMENT(APAR CAUSES A LOOP).

I                                     JBB1202  AZ00002  AZ00002  ++HOLD (UZ00001) FMID(JBB1202)
I                                     UZ00023*  ERROR REASON(AZ00002)
I                                     UZ12090   COMMENT(INCORRECT OUTPUT
I                                     DATA SET).

UZ00325  EJS2409  AZ07003  AZ07003  ++HOLD (UZ00325) FMID(EJS2409)
                                     EJS2410   ERROR REASON(AZ07003)
                                     COMMENT(SMPPTFIN NOT FOUND).

UZ64781  JXY0033  AZ07092  AZ07092  ++HOLD (UZ64781) FMID(JXY0033)
                                     UZ00012   ERROR REASON(AZ07092)
                                     UZ00023   COMMENT(APAR CAUSES AN
                                     UZ00204*   0C4 ABEND).
                                     UZ44401
                                     UZ78003
                                     AZ60098  ***NONE  ++HOLD (UZ64781) FMID(JXY0033)
                                     ERROR REASON(AZ60098)
                                     COMMENT(INDEFINITE WAIT
                                     STATE).
                                     AZ60099  UZ00260  ++HOLD (UZ64781) FMID(JXY0033)
                                     UZ90790   ERROR REASON(AZ60099)
                                     COMMENT(LOG DATA SET NOT
                                     AVAILABLE).
    
```

Figure 41. Exception SYSMOD Report: Sample Report

## File Allocation Report

This report is produced at the completion of each SMP/E command (except SET and RESETRC) to identify the DD statements used for that command. It shows how each DD statement was obtained and contains information about the DD statement. When filling in the report fields for a data set, SMP/E relies on the system information for that data set, which is stored in the job file control block (JFCB). The report is arranged alphanumerically by ddname. (Regardless of how the SMPTLIB data sets are allocated, they do not appear in the File Allocation report.)

If an error occurs before SMP/E has collected the necessary information about the command's DD statements, the report is not produced.

## Format and Explanation of Data

PAGE <i>nnnn</i> - NOW SET TO <i>zzzzzz</i> ZONE <i>nnnnnnn</i> DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT								
SMP/E <i>xxxxxxx</i> FILE ALLOCATION REPORT								
ZONE	DDNAME	DDDEFNAM	SMPDDNAM	TYPE	-----DATA SET OR PATH-----	VOLSER	UNIT	STATUS
<i>zzzz</i>	<i>aaaaaaa</i>	<i>bbbbbbb</i>		<i>cccc</i>	<i>dd</i>	<i>eeee</i>	<i>ffffff</i>	<i>ggggg</i>
	<i>aaaaaaa</i>	<i>bbbbbbb</i>	<i>lllllll</i>	<i>cccc</i>	<i>dd</i>	<i>eeee</i>	<i>ffffff</i>	<i>ggggg</i>
		<i>bbbbbbb</i>	<i>lllllll</i>	<i>cccc</i>	<i>dd</i>	<i>eeee</i>	<i>ffffff</i>	<i>ggggg</i>
		<i>bbbbbbb</i>	<i>lllllll</i>	<i>cccc</i>	<i>dd</i>	<i>eeee</i>	<i>ffffff</i>	<i>ggggg</i>
		<i>bbbbbbb</i>	<i>lllllll</i>	<i>cccc</i>	<i>dd</i>	<i>eeee</i>	<i>ffffff</i>	<i>ggggg</i>
	<i>aaaaaaa</i>	<i>bbbbbbb</i>		<i>cccc</i>	<i>dd</i>	<i>eeee</i>	<i>ffffff</i>	<i>ggggg</i>

Figure 42. File Allocation Report: Standard Format

These are the fields in the report:

**xxxxxxx**

is the SMP/E command being processed.

**ZONE**

is one of the following:

- For the APPLY and RESTORE commands, the ZONE field contains the name of the cross-zone whose DDDEFs have been used for the allocation of the library. An SMP-generated DD is used instead of the DDNAME to keep all currently allocated DDs unique.
- For the LINK command, the ZONE field contains the name of the FROMZONE or DLIB zone related to the FROMZONE whose DDDEF has been used to allocate the library.

**DDNAME**

is the ddname of the DD statement used. It can be either a user-specified ddname or one that was generated by SMP/E.

**DDDEFNAM**

is the name of the DDDEF entry that was used to allocate the DDNAME data set. This field is filled in only if SMP/E dynamically allocated the DD statement. It is blank for:

- Background processing when the JCL supplied a DD statement

- Foreground processing when the data set was preallocated using the TSO ALLOCATE command

The SMPPTS entry in the sample report is an example of a DD statement specified by the user (its DDDEFNAM field is blank).

### **SMPDDNAM**

is filled in only for concatenated DD statements that were dynamically allocated by SMP/E. When a DDDEF entry lists data sets to be concatenated, SMP/E dynamically allocates the individual members and assigns each data set a unique ddname. This name is shown in the SMPDDNAM field. SMP/E then requests that these individual ddnames be concatenated and assigned the ddname indicated in the DDNAME field. The SYSLIB entry in the sample report is an example of concatenated DD statements.

#### **Notes:**

1. For concatenated DD statements specified in the JCL, only the information from the first concatenated library is displayed.
2. This field is also used to show the data sets in a SYSLIB allocation for a load module, as indicated by the CALLLIBS subentry list in the corresponding LMOD entry. Each ddname in the CALLLIBS list is displayed, along with the information from its corresponding DDDEF entry and its SMP/E-generated ddname. When the CALLLIBS subentry list contains more than one name, SMP/E allocates them as a concatenation and gives the concatenation a generated ddname.

### **TYPE**

is either the type of data set that was allocated or the reason the data set was not allocated. It may be one of the following:

- ERROR** An error occurred when SMP/E tried to dynamically allocate the specified ddname. See the error messages in the SMPOUT data set to determine the cause of the error.
- NODDF** There was no DDDEF entry in the current zone; so SMP/E could not dynamically allocate the requested DD statement.
- The SMPRPT entry in the sample report is an example of a DD statement that was not found. In this case, because SMP/E could not allocate the SMPRPT DD statement, it had to write all reports to the SMPOUT DD statement.
- NTFND** The JCL did not contain the required DD statement. SMP/E tries to dynamically allocate the DD statement using the DDDEF entries.
- PERM** The DD statement specified a permanent data set.
- SYSIO** The DD statement specified a SYSIN or SYSOUT data set.
- The SMPOUT entry in the sample report is an example of a SYSIO data set. The data set name in this sample is the temporary JES2 data set name (shown as JES2.A.B...).
- TEMP** The DD statement specified a temporary data set.
- The entries for SMPWRK1 through SMPWRK6 in the sample report are examples of temporary data sets allocated by SMP/E.

**VIO** The DD statement specified is a VIO data set.

The entries for SYSUT1 through SYSUT4 in the sample report are examples of VIO data sets specified by the user; their DDDEFNAM fields are blank.

**DATA SET OR PATH**

is the name of the data set or path specified in the DDDEF entry or in the JCL.

- For data sets, the full data set name (up to 44 characters) is displayed.
- For paths, the full pathname (up to 255 characters) is displayed. The pathname is enclosed by apostrophes. If the pathname plus the enclosing apostrophes is greater than 44 characters, the pathname spans several lines.

**VOLSER**

identifies the volume specified in the DDDEF entry, the JCL, or the catalog.

**UNIT**

identifies the unit where the data set resides. This field is filled in if SMP/E dynamically allocated the DD statement and the DDDEF entry contained the unit information. Otherwise, this field is blank.

**STATUS**

is the status of the data set: NEW, OLD, MOD, or SHR.

Example: File Allocation Report for APPLY

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
SMP/E APPLY FILE ALLOCATION REPORT

```

ZONE	DDNAME	DDDEFNAM	SMPDDNAM	TYPE	-----DATA SET OR PATH-----	VOLSER	UNIT	STATUS
	BPXLIB1				'/etc/bin/'			
	BPXLIB2				'/etc/bin/this/is/an/example/of/a/pathname/t hat/is/>/44/characters/'			
	LINKLIB	LINKLIB		PERM	SYS1.LINKLIB		SYSRES	OLD
	SMPCNTL			SYSIO	JES2.A.B....			
	SMPLOG	SMPLOG		PERM	SYS1.SMPLOG		SYSRES	OLD
	SMPLTS	SMPLTS		PERM	SYS1.SMPLTS		SYSRES	SHR
	SMPMTS	SMPMTS		PERM	SYS1.SMPMTS		SYSRES	OLD
	SMPOUT	SMPOUT		SYSIO	JES2.A.B....			MOD
	SMPPTS			PERM	SYS1.SMPPTS		SMPVOL	OLD
	SMPRPT	SMPRPT		NODDF				
	SMPSCDS	SMPSCDS		PERM	SYS1.SMPSCDS		SYSRES	OLD
	SMPSTS	SMPSTS		PERM	SYS1.SMPSTS		SYSRES	OLD
	SMPWRK1	SMPWRK1		TEMP			SCR001	NEW
	SMPWRK2	SMPWRK2		TEMP			SCR001	NEW
	SMPWRK3	SMPWRK3		TEMP			SCR001	NEW
	SMPWRK4	SMPWRK4		TEMP			SCR001	NEW
	SMPWRK6	SMPWRK6		TEMP			SCR001	NEW
	SMP00001							
	BPXLIB3	SMP00001			'/etc/bin/calllib/'			
	SMP00002							
	PLIBASE	SMP00002		PERM	SYS1.PLIBASE			SHR
	CSSLIB	SMP00003		PERM	SYS1.CSSLIB			SHR
	SVCLIB	SVCLIB		PERM	SYS1.SVCLIB		SYSRES	OLD
	SYSLIB	SYSLIB						
	SMPMTS	SYSLIB		PERM	SYS1.SMPMTS		SYSRES	OLD
	MACLIB	SMP00004		PERM	SYS1.MACLIB		SYSRES	OLD
	AMACLIB	SMP00005		PERM	SYS1.AMACLIB		DLIB01	OLD
	SYSPRNT	SYSPRNT		SYSIO	JES2.A.B....			MOD
	YSUT1			VIO				NEW
	YSUT2			VIO				NEW
	YSUT3			VIO				NEW
	YSUT4			VIO				NEW
OPNMVS	SMP00006							
OPNMVS	BPXLIB1	SMP00006			'/etc/bin/cross/'			
XTZONE1	SMP00007							
XTZONE1	PLIBASE	SMP00007		PERM	SYS1.XTZONE1.PLIBASE			SHR
XTZONE1	CSSLIB	SMP00008		PERM	SYS1.XTZONE1.CSSLIB			SHR

Figure 43. File Allocation Report: Sample Report for APPLY

## GENERATE Summary Report

This report is produced during GENERATE processing to summarize the jobs that have been created.

### Format and Explanation of Data

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

GENERATE SUMMARY REPORT

JOBNAME  STEPNAME UTILITY  SYSLIB  MEMBER  TYPE    DISTLIB  MEMBER  FMID

aaaaaaaa bbbbbbbb cccccccc dddddddd eeeeeeee ffffffff gggggggg hhhhhhhh iiiiii
aaaaaaaa bbbbbbbb cccccccc dddddddd eeeeeeee ffffffff gggggggg hhhhhhhh iiiiii
aaaaaaaa bbbbbbbb cccccccc dddddddd eeeeeeee ffffffff gggggggg hhhhhhhh iiiiii
aaaaaaaa bbbbbbbb cccccccc dddddddd eeeeeeee ffffffff gggggggg hhhhhhhh iiiiii

```

Figure 44. GENERATE Summary Report: Standard Format

These are the fields in the report:

#### JOBNAME

is the job name that was generated. The job name is displayed once for the first step of each job. If the job is continued on another page of the report, the job name is repeated at the top of the new page.

#### STEPNAME

is the step name that was generated. Each step name is displayed only once. If the step is continued on another page of the report, the step name is repeated at the top of the new page.

#### UTILITY

is the name of the utility to be called. The utility program name is displayed once for each step.

#### SYSLIB

is the name of the target library that will be updated. The target library ddname is displayed at the start of each step and whenever it changes within that step.

#### MEMBER

is the member name of the element in the target library. The member name is displayed once for each member in the target library.

#### TYPE

is the element type of the SYSLIB member.

**Note:** ASSEM appears only for assembly steps, indicating that the source of the assembly input was the target zone ASSEM entry. If ASSEM is in the TYPE field, no DISTLIB value is displayed.

#### DISTLIB

is the name of the distribution library from which the element will be obtained. The DISTLIB field is displayed on each line (except for assembly lines for ASSEM entries).

#### MEMBER

is the member name of the element in the distribution library.

**FMID**

is the FMID of the function that owns the element.

**Examples**

The following sample reports are provided:

- “Example 1: No Load Modules with a SYSLIB Allocation”
- “Example 2: Load Modules with a SYSLIB Allocation” on page 469

**Example 1: No Load Modules with a SYSLIB Allocation**

This example shows the kind of information that is normally provided on the GENERATE Summary report.

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

GENERATE SUMMARY REPORT

JOBNAME  STEPNAME UTILITY  SYSLIB  MEMBER  TYPE    DISTLIB  MEMBER  FMID
-----  -
COPYJOB  COPYSTEP IEBCOPY  GENLIB  MAC01   MAC     AGENLIB  MAC01   PAA1100
                   MAC02   MAC     AGENLIB  MAC02   PAA1100
                   MACLIB  MAC11   MAC     AMACLIB  MAC11   PAA1100
                   MAC12   MAC     AMACLIB  MAC12   PAA1100
                   SRCLIB  SRC11   SOURCE  ASRCLIB  SRC11   PAA1100
                   SRC12   SOURCE  ASRCLIB  SRC12   PAA1100
                   LPALIB  MOD01   LMOD    AOS12    MOD01   PAA1100
                   MOD02   LMOD    AOS13    MOD02   PAA1100
                   ISRCLIB  ISRFC01 CLIST   AISRCLIB  ISRFC01 PAA1100
                   JPNHPLB  HELPK01 HELPJPN  AJPNHPLB  HELPK01 PAA1100
                   HELPK02  HELPJPN  AJPNHPLB  HELPK02  PAA1100
                   FRAMSLB  MSGFRA01 MSGFRA  AFRAMSLB  MSGFRA01 PAA1100
                   DEUMSLB  MSGDEU01 MSGDEU  ADEUMSLB  MSGDEU01 PAA1100
                   MSGDEU02 MSGDEU  ADEUMSLB  MSGDEU02 PAA1100
                   MSGDEU03 MSGDEU  ADEUMSLB  MSGDEU03 PAA1100
                   PARMLIB  PARMXX01 PARM    APARMLIB  PARMXX01 PAA1100
                   ISPLLIB  ISP@PRIM PNLENG  AIPPLIB  ISP@PRIM PAA1100
                   ITASPLB  SAMPIT07 SAMPITA AITASPLB  SAMPIT07 PAA1100
                   ESPTXLB  TXTESP66 TEXTESP  AESPTXLB  TXTESP66 PAA1100
LINKLIB  ASSM0001 IEUASM  ASSEM01  ASSEM    ASSEM01  PAA1100
                   ASSM0002 IEUASM  ASSEM02  ASSEM    ASSEM02  PAA1100
                   ASSM0003 IEUASM  SRCMOD01 SOURCE  SRCDLIB  SRCMOD01 PAA1100
                   ASSM0004 IEUASM  SRCMOD02 SOURCE  SRCDLIB  SRCMOD02 PAA1100
LINK0001 IEWL  LINKLIB  LMOD01   LMOD     AOS12    MOD21   PAA1100
                   AOS12    MOD22   PAA1100
                   AOS12    MOD23   PAA1100
                   LINK0002 IEWL  LINKLIB  LMOD02   LMOD     SMPPUNCH ASSEM01  PAA1100
                   SMPPUNCH ASSEM02  PAA1100
                   SMPPUNCH SRCMOD01  PAA1100
                   SMPPUNCH SRCMOD02  PAA1100
SVCLIB  LINK0001 IEWL  SVCLIB  LMOD50   LMOD     DN554    MOD50   PAA1100
                   LMOD51   LMOD     DN554    MOD51   PAA1100
    
```

Figure 45. GENERATE Summary Report: Sample Report



**Example 2: Load Modules with a SYSLIB Allocation**

This example shows the kind of information that is provided on the GENERATE Summary report when load modules have a SYSLIB allocation (the LMOD entry contains a CALLLIBS subentry list). The SMPLTS job link-edits the base version of such load modules into the SMPLTS data set, and the LKSYSLIB job link-edits the executable version of the load modules into the target libraries.

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

GENERATE SUMMARY REPORT

JOBNAME  STEPNAME  UTILITY  SYSLIB  MEMBER  TYPE  DISTLIB  MEMBER  FMID
COPYJOB  COPYSTEP  IEBCOPY  SRCLIB  MSAPL03 SOURCE DSRCLIB  MSAPL03 FGNSRC
LPALIB   LINK0001  IEWL     LPALIB  FTNLMOD1 LMOD  DLIB1    FTNMOD1 FGENFTN
                                                DLIB1    FTNMOD2 FGENFTN
SMPLTS   ASSM0001  IEV90    ASSEM   MSAPL01
        ASSM0002  IEV90    ASSEM   MSAPL02
        LINK0001  IEWL     SMPLTS  LMOD3   LMOD  DLIB1    MOD1     FGEN001
                                                DLIB1    MOD2     FGEN001
                                                DLIB1    MOD3     FGEN001
                                                SMPLTS  LMOD4   LMOD  DLIB1    MOD1     FGEN001
                                                DLIB1    MOD2     FGEN001
                                                DLIB1    MOD3     FGEN001
        LINK0002  IEWL     SMPLTS  LMOD1   LMOD  DLIB1    MOD1     FGEN001
                                                DLIB1    MOD2     FGEN001
                                                DLIB1    MOD3     FGEN001
        SMPLTS  LMOD2   LMOD  DLIB1    MOD1     FGEN001
                                                DLIB1    MOD2     FGEN001
                                                DLIB1    MOD3     FGEN001
        SMPLTS  PLILMOD1 LMOD  DLIB1    PLIMOD1  FGENPL1
        LINK0003  IEWL     SMPLTS  LAPLsrc1 LMOD  DLIB1    PLIMOD2  FGENPL1
                                                SYSPUNCH MSAPL01
                                                SYSPUNCH MSAPL02
LKSYSLIB ASSM0001  IEV90    ASSEM   MSAPL03 FGNSRC
        ASSM0002  IEV90    ASSEM   MSAPL01
        LINK0001  IEWL     CSSLIB  CSSLMOD1 LMOD  DLIB1    CSSMOD1  FGEN001
                                                DLIB1    CSSMOD2  FGEN001
                                                DLIB1    CSSMOD3  FGEN001
        CSSLIB  CSSLMOD2 LMOD  DLIB1    CSSMOD1  FGEN001
                                                DLIB1    CSSMOD2  FGEN001
        LINK0002  IEWL     FORTLIB FTNLMOD1 LMOD  DLIB1    FTNMOD1  FGENFTN
                                                DLIB1    FTNMOD2  FGENFTN
        LINK0003  IEWL     LINKLIB  LMOD1   LMOD  DLIB1    MOD1     FGEN001
                                                DLIB1    MOD2     FGEN001
                                                DLIB1    MOD3     FGEN001
        LINKLIB  LMOD2   LMOD  DLIB1    MOD1     FGEN001
                                                DLIB1    MOD2     FGEN001
                                                DLIB1    MOD3     FGEN001
        LINK0004  IEWL     LINKLIB  LAPLsrc1 LMOD  SYSPUNCH MSAPL01
                                                SYSPUNCH MSAPL02
                                                DLIB1    MSAPL03  FGNSRC
    
```

Figure 46 (Part 1 of 2). GENERATE Summary Report: Sample Report for LMODs with Multiple FMIDs

# GENERATE Summary Report

LINK0005	IEWL	LPALIB	LMOD3	LMOD	DLIB1	MOD1	FGEN001	
					DLIB1	MOD2	FGEN001	
					DLIB1	MOD3	FGEN001	
		LPALIB	LMOD4	LMOD	DLIB1	MOD1	FGEN001	
					DLIB1	MOD2	FGEN001	
					DLIB1	MOD3	FGEN001	
LINK0006	IEWL	LPALIB	LMOD1	LMOD	DLIB1	MOD1	FGEN001	
					DLIB1	MOD2	FGEN001	
					DLIB1	MOD3	FGEN001	
		LPALIB	LMOD2	LMOD	DLIB1	MOD1	FGEN001	
					DLIB1	MOD2	FGEN001	
					DLIB1	MOD3	FGEN001	
LINK0007	IEWL	LPALIB	LAPL SRC1	LMOD	SYSPUNCH	MSAPL01		
					SYSPUNCH	MSAPL02		
					DLIB1	MSAPL03	FGENSRC	
LINK0008	IEWL	PLILIB	PLILMOD1	LMOD	DLIB1	PLIMOD1	FGENPL1	
					DLIB1	PLIMOD2	FGENPL1	
HFSINST	SLIB04	GIMGNIAP	SLIB04	ELEM1	HFS	DLIB04	ELEM1	FUNC001
			SLIB04	ELEM10	HFS	DLIB04	ELEM10	FUNC003
			SLIB04	ELEM11	HFS	DLIB04	ELEM11	FUNC004
			SLIB04	ELEM2	HFS	DLIB04	ELEM2	FUNC001

Figure 46 (Part 2 of 2). GENERATE Summary Report: Sample Report for LMODs with Multiple FMIDs

## JCLIN Cross-Reference Report

This report is produced during JCLIN processing to show where to place specific entries in the JCLIN Summary report.

For inline JCLIN processing, the report title line is:

```
JCLIN CROSS REFERENCE REPORT FOR SYSMOD nnnnnnnn
```

where *nnnnnnnn* is the ID of the SYSMOD containing the inline JCLIN. Several of these reports can be produced during APPLY or ACCEPT, one for each SYSMOD with inline JCLIN.

**Note:** If SMP/E runs out of storage during JCLIN processing, the JCLIN Cross-Reference report may not be produced.

## Format and Explanation of Data

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

JCLIN CROSS REFERENCE REPORT

__TYPE__ __NAME__ __WHERE USED__ __TYPE__ __NAME__ __WHERE USED__

aaaaaaaa bbbbbbbb nnnn nnnn nnnn aaaaaaaaa bbbbbbbb nnnn nnnn nnnn
aaaaaaaa bbbbbbbb nnnn nnnn nnnn aaaaaaaaa bbbbbbbb nnnn nnnn nnnn
aaaaaaaa bbbbbbbb nnnn nnnn nnnn aaaaaaaaa bbbbbbbb nnnn nnnn nnnn
aaaaaaaa bbbbbbbb nnnn nnnn nnnn aaaaaaaaa bbbbbbbb nnnn nnnn nnnn
aaaaaaaa bbbbbbbb nnnn nnnn nnnn aaaaaaaaa bbbbbbbb nnnn nnnn nnnn
```

Figure 47. JCLIN Cross-Reference Report: Standard Format

These are the fields in the report:

### TYPE

is the entry type. The possible entry types, listed in the order in which they appear, are:

- ASSEM
- MAC
- LMOD
- MOD
- SRC
- DLIB

### NAME

is the entry name.

### WHERE USED

is a list of the pages in the JCLIN Summary report where changes for the entry are noted.

### Example: JCLIN Cross-Reference Report

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

      JCLIN CROSS REFERENCE REPORT

__TYPE__ __NAME__ __WHERE USED__ __TYPE__ __NAME__ __WHERE USED__

MOD      GIMMPGTA 0001 0003 0004
          0010 0015 0016
          0100
MOD      GIMMPGTB 0001 0003 0004
          0010 0015 0016
          0100
MOD      GIMMPG01 0001 0003
MOD      GIMMPG02 0001 0003
MOD      GIMMPG03 0001 0003
MOD      GIMMPG04 0001 0003
```

Figure 48. JCLIN Cross-Reference Report: Sample Report

## JCLIN Summary Report

This report is produced during JCLIN processing to summarize the changes that have been made.

For inline JCLIN processing, the report title line is:

JCLIN SUMMARY REPORT FOR SYSMOD *nnnnnnnn*

where *nnnnnnnn* is the ID of the SYSMOD containing the inline JCLIN. Several of these reports can be produced during APPLY or ACCEPT, one for each SYSMOD with inline JCLIN.

## Format and Explanation of Data

PAGE <i>nnnn</i> - NOW SET TO <i>zzzzzz</i> ZONE <i>nnnnnnn</i> DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT							
JCLIN SUMMARY REPORT							
_____ JCL INFORMATION _____				_____ SMP/E ACTION TAKEN _____			
JOBNAME	STEPNAME	UTILITY	TYPE	MEMBER	TYPE	ACTION	CHANGES MADE
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>cccccccc</i>	<i>ddd</i>	<i>eeeeeee</i>	<i>fff</i>	<i>gggggg</i>	<i>hhhhhhhhhh</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>cccccccc</i>	<i>ddd</i>	<i>eeeeeee</i>	<i>fff</i>	<i>gggggg</i>	<i>hhhhhhhhhh</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>cccccccc</i>	<i>ddd</i>	<i>eeeeeee</i>	<i>fff</i>	<i>gggggg</i>	<i>hhhhhhhhhh</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>cccccccc</i>	<i>ddd</i>	<i>eeeeeee</i>	<i>fff</i>	<i>gggggg</i>	<i>hhhhhhhhhh</i>

Figure 49. JCLIN Summary Report: Standard Format

These are the fields in the report:

### JOBNAME

is the name of the current job being analyzed. The job name is displayed once for the first step of each job. If the job is continued on another page of the report, the job name is repeated at the top of the new page.

### STEPNAME

is the name of the current step being analyzed. Each step name is displayed only once. If the step is continued on another page of the report, the step name is repeated at the top of the new page.

### UTILITY

is the name of the utility program or catalogued procedure being analyzed. The utility program name is displayed once for each step.

### TYPE

is the utility type, which can be:

- ASSM** Assembly step
- COPY** Copy step
- LINK** Link-edit step
- UNKN** Ignored step (the utility was not recognized)
- UPDT** Update step

### MEMBER

is the name of the entry that was modified. The entry name is displayed once for each changed entry.

## TYPE

is the entry type: ASSEM, DLIB, LMOD, MAC, MOD, or SRC.

## ACTION

is the type of action taken:

**ADDED** No entry existed, but one was added.

**UPDATED** An existing entry was updated.

**DELETED** An existing entry was deleted.

## CHANGES MADE

describes the type of changes made to the entry, which may be one or more of the following. `xxxxxxx` is the value of the field, and `yyyyyyyy` is the new value of the field if it was changed.

- NO UPDATES REQUIRED

This can appear for all types of entries and JCLIN steps. It indicates that the same JCLIN was previously processed, and because there were no changes in the JCLIN, SMP/E did not need to change any entries as a result of processing the JCLIN. Here are some instances when this might occur:

- A PTF included the entire product JCLIN, not just the small part that was changed or added. In this case, most of the entries need no updates, because there is no change to the related JCLIN.
- An APPLY step completed JCLIN processing (so the entries were updated with the JCLIN changes), but a library ran out of space during subsequent processing of that same APPLY step. When the APPLY step is rerun, no JCLIN updates are necessary, because the entries have already been updated.

- ASSEM entry:

- ASSEMBLER INPUT ADDED
- ASSEMBLER INPUT REPLACED

- DLIB entry:

- SYSLIB=`xxxxxxx`
- SYSLIB `xxxxxxx` ADDED
- SYSLIB `xxxxxxx` REPLACED BY `yyyyyyyy`

- LMOD entry:

- CALLLIBS ADDED
- CALLLIBS REPLACED
- COPY INDICATOR SET
- COPY INDICATOR DELETED
- LEPARMS ADDED
- LEPARMS REPLACED
- LINK-EDIT INPUT ADDED
- LINK-EDIT INPUT REPLACED
- LINK-EDIT INPUT DELETED
- NOT PROCESSED
- SYSLIB=`xxxxxxx`
- SYSLIB `xxxxxxx` ADDED
- SYSLIB `xxxxxxx` REPLACED BY `yyyyyyyy`

**Note:** When SMPLTS is the output library, JCLIN link-edit steps are skipped.

- MAC entry:
  - DISTLIB=xxxxxxxx
  - DISTLIB xxxxxxxx ADDED
  - DISTLIB xxxxxxxx REPLACED BY yyyyyyyy
  - GENASM=xxxxxxxx
  - GENASM xxxxxxxx ADDED
  - MALIAS=xxxxxxxx
  - MALIAS xxxxxxxx ADDED
  - SYSLIB=xxxxxxxx
  - SYSLIB xxxxxxxx ADDED
  - SYSLIB xxxxxxxx REPLACED BY yyyyyyyy
- MOD entry:
  - DISTLIB=xxxxxxxx
  - DISTLIB xxxxxxxx ADDED
  - DISTLIB xxxxxxxx REPLACED BY yyyyyyyy
  - LMOD=xxxxxxxx
  - LMOD xxxxxxxx ADDED
  - LMOD xxxxxxxx DELETED
  - TALIAS=xxxxxxxx
  - TALIAS xxxxxxxx ADDED
- SRC entry:
  - DISTLIB=xxxxxxxx
  - DISTLIB xxxxxxxx ADDED
  - DISTLIB xxxxxxxx REPLACED BY yyyyyyyy
  - SYSLIB=xxxxxxxx
  - SYSLIB xxxxxxxx ADDED
  - SYSLIB xxxxxxxx REPLACED BY yyyyyyyy

Example: JCLIN Summary Report

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

      JCLIN SUMMARY REPORT

_____ JCL INFORMATION _____ SMP/E ACTION TAKEN _____
JOBNAME  STEPNAME  UTILITY  TYPE  MEMBER  TYPE  ACTION  CHANGES MADE
JOB001   STEP001   IEWL     LINK  LMOD001  LMOD  ADDED   SYSLIB=LINKLIB,
                                         LEPARMS ADDED,
                                         LINK-EDIT INPUT
                                         ADDED
                                         MOD0001  MOD  ADDED   DISTLIB=AOS12,
                                         LMOD=LMOD001
                                         MOD0002  MOD  ADDED   DISTLIB=AOS12,
                                         LMOD=LMOD001
                                         LMOD002  LMOD ADDED   SYSLIB=LINKLIB,
                                         LEPARMS ADDED,
                                         LINK-EDIT INPUT
                                         ADDED
                                         MOD0004  MOD  ADDED   DISTLIB=AOS12,
                                         LMOD=LMOD001
                                         MOD0005  MOD  ADDED   DISTLIB=AOS12,
                                         LMOD=LMOD001
      STEP002  IEBCOPY  COPY  MAC001  MACRO  ADDED   DISTLIB=AMACLIB,
                                         SYSLIB=MACLIB
                                         MAC002  MACRO  ADDED   DISTLIB=AMACLIB,
                                         SYSLIB=MACLIB
      STEP003  IDCAMS  UNKN
      STEP004  IEUASM  ASSM   NO UPDATES REQUIRED
JOB002   STEP001  IEUASM  ASSM  ASSEM001 ASSEM  UPDATED ASSEMBLER INPUT
                                         REPLACED
                                         MAC001  MACRO  UPDATED GENASM ASSEM001
                                         ADDED
                                         MAC002  MACRO  UPDATED GENASM ASSEM001
                                         ADDED
      STEP002  IEBCOPY  COPY  AMACLIB  DLIB  ADDED   SYSLIB=MACLIB
      STEP003  IEBCOPY  COPY  MOD0005  MOD  UPDATED DISTLIB AOS12
                                         REPLACED BY
                                         ALINKLIB
    
```

Figure 50. JCLIN Summary Report: Sample Report



## LIST Summary Report

This report is produced during LIST processing to summarize which entries were or were not found in the set-to zone.

### Format and Explanation of Data

PAGE <i>nnnn</i> - NOW SET TO <i>zzzzzz</i> ZONE <i>nnnnnnn</i> DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT		
LIST SUMMARY FOR <i>aaaaaaaa</i>		
ENTRY-TYPE	ENTRY-NAME	STATUS
<i>bbbbbbbb</i>	<i>ccccccc</i>	<i>ddddddddddddddd</i>
<i>bbbbbbbb</i>	<i>ccccccc</i>	<i>ddddddddddddddd</i>
<i>bbbbbbbb</i>	<i>ccccccc</i>	<i>ddddddddddddddd</i>
<i>bbbbbbbb</i>	<i>ccccccc</i>	<i>ddddddddddddddd</i>

Figure 51. LIST Summary Report: Standard Format

These are the fields in the report:

***aaaaaaaa***

is the name of the zone containing the entries being listed.

#### **ENTRY-TYPE**

is the type of entry SMP/E looked for. If no specific entries of a given type were selected, that entry type is shown only once. If several specific entries of a given type were selected, that entry type is shown for each of the selected entries.

#### **ENTRY-NAME**

is the name of an entry that was specified on the LIST command. If no specific entries were selected, this is blank.

#### **STATUS**

indicates whether any entries were found. The status can be one of the following:

##### **NOT FOUND**

The specified entry or entry type was not found.

##### **STARTS ON PAGE *nnnn***

The specified entry or entry type was found. The LIST output starts on the indicated page in the SMPLIST data set.

##### **EMPTY ZONE—NO ENTRIES FOUND**

This may appear for the LIST or LIST ALLZONES command. No entries were found in the specified zone. This is the only line in the report. The entry type and entry name fields are blank.

## NO ENTRIES FOUND

This may appear for the **LIST FORFMID**(*name*) or **LIST BACKUP**(*sysmod\_id*) command. No entries were found for the indicated FORFMID or SYSMOD ID values in the specified zone. This is the only line in the report. The entry type and entry name fields are blank.

## Example: LIST Summary Report

Figure 52 shows the LIST Summary report that can accompany the LIST output for the following command:

**LIST MOD MAC SRC CLIST(CLIST01,CLIST02,CLIST03).**

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
LIST SUMMARY FOR MVSTGT1
ENTRY-TYPE  ENTRY-NAME  STATUS
CLIST       CLIST01    STARTS ON PAGE 0002
CLIST       CLIST02    NOT FOUND
CLIST       CLIST03    STARTS ON PAGE 0002
MAC
MOD         STARTS ON PAGE 0001
SRC         STARTS ON PAGE 0010
```

Figure 52. LIST Summary Report: Sample Report

## MOVE/RENAME/DELETE Report

This report is produced when SMP/E has processed a SYSMOD containing ++MOVE, ++RENAME, or ++DELETE statements. It can be written for the following commands: ACCEPT, ACCEPT CHECK, APPLY, APPLY CHECK, RESTORE, and RESTORE CHECK.

### Format and Explanation of Data

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT				
SMP/E MOVE/RENAME/DELETE REPORT FOR xxxxxxxx PROCESSING				
ELEMENT TYPE	ELEMENT NAME	SYSMOD NAME	ACTION	COMMENT
aaaaaaaa	bbbbbbbb	cccccc	dd	ee
aaaaaaaa	bbbbbbbb	cccccc	dd	ee
aaaaaaaa	bbbbbbbb	cccccc	dd	ee
aaaaaaaa	bbbbbbbb	cccccc	dd	ee
aaaaaaaa	bbbbbbbb	cccccc	dd	ee

Figure 53. MOVE/RENAME/DELETE Report: Standard Format

These are the fields in the report:

xxxxxxx  
is the command being processed:

- ACCEPT
- ACCEPT CHECK
- APPLY
- APPLY CHECK
- RESTORE
- RESTORE CHECK

**ELEMENT TYPE**  
is the type of element that was changed: MAC, MOD, SRC, or LMOD.

**ELEMENT NAME**  
is the name of an element that was changed.

**SYSMOD NAME**  
identifies the SYSMOD containing the MCS that changed the element.

**ACTION**  
is the type of change that was made:

**ALIAS DELETED FROM SYSLIB** *syslib - alias*  
The load module alias was deleted from the indicated library during APPLY processing.

**Notes:**

1. A long alias spans several lines.
2. An alias is enclosed in apostrophes if it contains a character other than upper case alphabetic, numeric, national (\$, #, or @), slash, plus, hyphen, period, and ampersand, or it spans several lines.

**ALIAS NOT DELETED FROM SYSLIB *syslib* - *alias***

The load module alias was not deleted from the indicated library during APPLY processing.

**Notes:**

1. A long alias spans several lines.
2. An alias is enclosed in apostrophes if it contains a character other than upper case alphabetic, numeric, national (\$, #, or @), slash, plus, hyphen, period, and ampersand, or it spans several lines.

**CHANGED DISTLIB FROM *distlib1* TO *distlib2***

The DISTLIB subentry of the element entry was changed during APPLY or RESTORE processing.

**CHANGED SYSLIB FROM *syslib1* TO *syslib2***

The SYSLIB subentry of the element entry was changed during ACCEPT processing.

**DELETED FROM SYSLIB *syslib1***

The load module was deleted from the indicated library during APPLY processing.

**DISTLIB NOT CHANGED FROM *distlib1* TO *distlib2***

The DISTLIB subentry of the element was not changed during APPLY or RESTORE processing.

**LMOD ENTRY NOT RENAMED TO *newname***

The indicated LMOD entry was not renamed during ACCEPT processing.

**LMOD ENTRY RENAMED TO *newname***

The indicated LMOD entry was renamed during ACCEPT processing.

**MOD *aaaaaaaa* IN ZONE *bbbbbbb* UPDATED**

The indicated cross-zone module is part of a load module renamed by the ++RENAME statement during APPLY or RESTORE processing. The cross-zone MOD entry has been updated to reflect the new load module name.

**MOD *aaaaaaaa* IN ZONE *bbbbbbb* NOT UPDATED**

The indicated cross-zone module is part of a load module renamed by the ++RENAME statement during APPLY or RESTORE processing. The cross-zone MOD entry was **not** updated to reflect the new load module name.

**MOVED FROM DISTLIB *distlib1* TO *distlib2***

The element was moved to a different distribution library during ACCEPT processing.

**MOVED FROM SYSLIB *syslib1* TO *syslib2***

The element was moved to a different target library during APPLY or RESTORE processing.

**NOT DELETED FROM SYSLIB *syslib1***

The load module was not deleted from the indicated library during APPLY processing.

**NOT MOVED FROM DISTLIB *distlib1* TO *distlib2***

The element was not moved during ACCEPT processing.

**NOT MOVED FROM SYSLIB *syslib1* TO *syslib2***

The element or load module was not moved during APPLY or RESTORE processing.

**NOT RENAMED TO *newname* IN SYSLIB *syslib1***

The load module was not renamed in the indicated library during APPLY or RESTORE processing.

**RENAMED TO *newname* IN SYSLIB *syslib1***

The load module was renamed in the indicated library during APPLY or RESTORE processing.

**SYSLIB NOT CHANGED FROM *syslib1* TO *syslib2***

The SYSLIB subentry of the element was not changed during ACCEPT processing.

**SYSLIB *syslib1* DELETED FROM LMOD ENTRY**

The SYSLIB subentry was deleted from the indicated LMOD entry during ACCEPT processing.

**SYSLIB *syslib1* NOT DELETED FROM LMOD ENTRY**

The SYSLIB subentry was not deleted from the indicated LMOD entry during ACCEPT processing.

**COMMENT**

provides additional information about processing:

**ALIAS(ES) *alias...***

For ++DELETE processing, the indicated aliases were deleted from the target library. For ++MOVE processing, they were moved to the new target library.

**Notes:**

1. A long alias spans several lines.
2. Multiple aliases are separated by a comma followed by a blank (" , ").
3. An alias is enclosed in apostrophes if it contains a character other than upper case alphabetic, numeric, national (\$, #, or @), slash, plus, hyphen, period, and ampersand, or it spans several lines.

**ALIAS GREATER THAN 8 CHARACTERS**

The ++DELETE statement attempted to delete an alias without deleting the associated load module, which resides in a PDSE. Because the alias to be deleted was more than 8 characters long, it could not be deleted.

**ALIAS NOT IN SYSLIB**

The alias to be deleted was not in the indicated target library.

**ALREADY CHANGED**

The DISTLIB or SYSLIB subentries for the element or load module has already been changed.

### **ALREADY DELETED**

The indicated load module has already been deleted.

### **ALREADY MOVED**

The element or load module has already been moved.

### **ALREADY RENAMED**

The indicated load module has already been renamed.

### **ATTEMPTED UPDATE FAILED**

Because of errors during cross-zone processing, the attempted cross-zone update was not made.

### **COPY FAILED**

The copy for a load module or element failed during ++MOVE processing.

### **CSI RESOURCE UNAVAILABLE**

Cross-zone processing was not done, because SMP/E could not obtain access to the CSI data set containing the cross-zone.

### **DEFERRED XZLINK SPECIFIED**

Cross-zone processing was not done, because the XZLINK value of the cross-zone is DEFERRED.

### **FMID MISMATCH**

The FMID of the element to be moved must match either the FMID of the SYSMOD containing the ++MOVE MCS, or an FMID specified on the ++MOVE MCS.

### **LIBRARIES NOT AVAILABLE**

Libraries needed for ++MOVE, ++RENAME, or ++DELETE processing were not available.

### **LINK EDIT FAILED**

The link-edit for a load module or element failed during ++MOVE processing.

### **MISSING ALIAS(ES) *alias...***

The indicated aliases were missing from the target or distribution library, even though the element or LMOD entry indicated that they should exist.

#### **Notes:**

1. A long alias spans several lines.
2. Multiple aliases are separated by a comma followed by a blank (" , ").
3. An alias is enclosed in apostrophes if it contains a character other than upper case alphabetic, numeric, national (\$, #, or @), slash, plus, hyphen, period, and ampersand, or it spans several lines.

### **MOD DOES NOT CONTAIN XZLMOD**

The LMOD from the set-to zone has an XZMOD subentry indicating that it contains the cross-zone module. SMP/E cannot update the cross-zone module's XZLMOD record to show that the LMOD was renamed, because the cross-zone module does **not** contain an XZLMOD subentry for the renamed LMOD.

If the LMOD does contain the cross-zone module, use UCLIN to add an XZLMOD subentry for the renamed LMOD to the cross-zone MOD entry. If the LMOD does not contain the cross-zone module, use UCLIN to remove the XZMOD subentry for the cross-zone module from the LMOD entry.

**Note:** If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

**MOD ENTRIES UPDATED**

For ++DELETE processing, the LMOD subentries in the MOD entries have been changed and the LMOD entry has been deleted. For ++RENAME processing, the LMOD subentries in the MOD entries have been changed.

**NEWNAME ALREADY EXISTS**

An LMOD entry already exists (as either an LMOD entry or in one of the LMOD's SYSLIBs) for the new name specified on a ++RENAME statement.

**NOT FOUND**

No entry was found for the element or load module named.

**NOT IN ALL SYSLIBS**

The load module was not in all the target libraries indicated in the LMOD entry.

**NOT IN CURRENT DISTLIB**

The element was not in the distribution library named. The DISTLIB on the MCS does not match the DISTLIB subentry in the element entry.

**NOT IN CURRENT SYSLIB**

The element or load module was not in the target library named. The SYSLIB on the MCS does not match the SYSLIB subentry in the element or LMOD entry.

**NOT INSTALLED**

The element or load module named was not installed or being installed.

**PREVIOUS ERROR**

A previous error prevented SMP/E from updating the cross-zone MOD XZLMOD subentry to indicate that the set-to LMOD was renamed. Use UCLIN to update the XZLMOD subentry for the renamed LMOD.

**SYSLIB SUBENTRY NOT FOUND**

The SYSLIB subentry of the element was not changed, because the SYSLIB subentry was not found in the element entry. This situation is common during ACCEPT processing for elements that are packaged in a totally copied library.

**SYSMOD TERMINATED**

The SYSMOD containing the ++MOVE, ++RENAME, or ++DELETE statement previously failed.

**XZLMOD SUBENTRY REPLACED**

Cross-zone processing has been successful and the MOD entry has been updated with the load module name.

**Note:** The report is limited to reporting on the first 20 aliases processed for a particular element. For example, suppose a load module having 23 aliases is being moved by a ++MOVE statement. The first 20 aliases moved appear in the MOVE/RENAME/DELETE report.

**Example: Report for APPLY Processing**

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
      SMP/E MOVE/RENAME/DELETE REPORT FOR APPLY PROCESSING
ELEMENT  ELEMENT  SYSMOD
TYPE     NAME     NAME     ACTION                                COMMENT
-----
MAC      GIMMPIO  UR06789  MOVED FROM SYSLIB MACLIB TO PTVMACS
          GIMMPIO  UR06789  CHANGED DISTLIB FROM AMACLIB TO APVTMACS
SRC      GIMMPSR  UR01234  MOVED FROM SYSLIB SRCLIB TO NEWSRC
LMOD    GIMSMP   UR04321  MOVED FROM SYSLIB LINKLIB TO PVTLIB
LMOD    GIMSMP1  UR06420  RENAMED TO GIMTEST IN SYSLIB LINKLIB
          GIMSMP1  UR06420  RENAMED TO GIMTEST IN SYSLIB PVTLIB
LMOD    GIMSMP2  UR09988  DELETED FROM SYSLIB LINKLIB
          GIMSMP2  UR09988  DELETED FROM SYSLIB PVTLIB
LMOD    GXXLMOD  UZ00442  MOVED FROM SYSLIB HFSPH1 TO HFSPH2
          GXXLMOD  UZ00442  ALIAS(ES) '../Friendly_altername_
          GXXLMOD  UZ00442  ame1_which_is_64_characters_long
          GXXLMOD  UZ00442  _exactly.', '../The_other_alterna
          GXXLMOD  UZ00442  e_is_SHORTER!!!!'
LMOD    HFSLMOD  UZ00440  ALIAS DELETED FROM SYSLIB HFSPH1 - '../I_am_a_64_cha
          HFSLMOD  UZ00440  racter_name,_wonderfully_friendly!_Think_so????'
          HFSLMOD  UZ00440  ALIAS DELETED FROM SYSLIB HFSPH2 - '../I_too_am_a_lon
          HFSLMOD  UZ00440  g_name._But_not_64_characters.'
LMOD    HFSLMOD  UZ00441  ALIAS NOT DELETED FROM SYSLIB HFSPH1 - '../lo'
          HFSLMOD  UZ00441  ALIAS NOT DELETED FROM SYSLIB HFSPH2 - ../OK
LMOD    LMODA    UR08856  NOT MOVED FROM SYSLIB LINKLIB TO LPALIB
LMOD    LMODC    UR08822  NOT MOVED FROM SYSLIB LINKLIB TO LPALIB
LMOD    LMODD    UR08544  NOT DELETED FROM SYSLIB LINKLIB
          LMODD    UR08544  NOT DELETED FROM SYSLIB PVTLIB
LMOD    LMODH    UR06455  ALIAS DELETED FROM SYSLIB LINKLIB - LOTTO
          LMODH    UR06455  ALIAS DELETED FROM SYSLIB PVTLIB - LOTTO
LMOD    LMODM    UR06455  ALIAS NOT DELETED FROM SYSLIB LINKLIB - LOTTO2
          LMODM    UR06455  ALIAS NOT DELETED FROM SYSLIB PVTLIB - LOTTO2
LMOD    LRMFD1   PREX014  NOT DELETED FROM SYSLIB HFSPH1
          LRMFD1   PREX014  MISSING ALIAS(ES) '../alternate_na
          LRMFD1   PREX014  me1_which_is_friendly_but_long',
          LRMFD1   PREX014  '../The_other_altername_is_SHORTER
          LRMFD1   PREX014  !!!!'
    
```

Figure 54. MOVE/RENAME/DELETE Report: Sample Report



## RECEIVE Exception SYSMOD Data Report

This report is produced at the completion of RECEIVE processing and shows the ++HOLD and ++RELEASE statements that were processed. This report is arranged by SYSMOD ID, and under each SYSMOD by category (ERROR first, SYSTEM second, and USER third), and under each category by reason ID. The ++HOLD text is displayed exactly as it is entered on the MCSs.

### Format and Explanation of Data

PAGE <i>nnnn</i> - NOW SET TO <i>zzzzzz</i> ZONE <i>nnnnnnn</i> DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT					
RECEIVE ++HOLD/++RELEASE SUMMARY REPORT					
NOTE: SMD NF - SYSMOD NOT RELEASED - NOT FOUND IN GLOBAL ZONE					
RSN NF - SYSMOD NOT RELEASED - NOT HELD FOR THIS REASONID					
INT HLD - SYSMOD NOT RELEASED - CANNOT RELEASE INTERNAL SYS HOLD					
SYSMOD	TYPE	STATUS	REASON	FMID	++HOLD MCS STATEMENTS
<i>aaaaaaa</i>	<i>bbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeee</i>	<i>ffffffffffffffffffff</i>
<i>aaaaaaa</i>	<i>bbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeee</i>	<i>ffffffffffffffffffff</i>
<i>aaaaaaa</i>	<i>bbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeee</i>	<i>ffffffffffffffffffff</i>
<i>aaaaaaa</i>	<i>bbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeee</i>	<i>ffffffffffffffffffff</i>
<i>aaaaaaa</i>	<i>bbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeee</i>	<i>ffffffffffffffffffff</i>

Figure 55. RECEIVE Exception SYSMOD Data Report: Standard Format

These are the fields in the report:

#### SYSMOD

identifies the SYSMOD specified on the ++HOLD or ++RELEASE statement.

#### TYPE

is the type of ++HOLD or ++RELEASE:

**ERR** Error hold  
**SYS** System hold  
**USER** User hold

#### STATUS

is the status of the ++HOLD or ++RELEASE statement. It can be one of the following:

**EXCLUDED** The SYSMOD was not processed, because it was specified on the EXCLUDE operand of the RECEIVE command.

**HELD** The ++HOLD statement was processed.

**N/A** The SYSMOD specified in the ++HOLD statement was not applicable to your system, because the FMID specified was not in the GLOBALZONE FMID list.

**RELEASED** The reason ID specified on the ++RELEASE statement was removed from the SYSMOD.

**SMD NF** SMP/E could not find a SYSMOD entry for the SYSMOD specified on the ++RELEASE statement.

- RSN NF** SMP/E could not find the reason ID specified on the ++RELEASE statement.
- INT HLD** The reason ID specified was for an internal SYSTEM HOLD, and therefore could not be released. You can resolve this type of HOLD only by using BYPASS(HOLDSYS) on ACCEPT or APPLY.

## REASON

identifies the SYSMOD specified in the REASON operand of the ++HOLD or ++RELEASE statement.

## FMID

is the FMID specified on the ++HOLD statement. For ++RELEASE statements this field is blank.

## ++HOLD MCS STATEMENTS

lists the complete text of the ++HOLD statements. For ++RELEASE statements this field is blank.

## Examples

The following sample reports are provided:

- “Example 1: Report for Internal HOLDDATA”
- “Example 2: Report for ++HOLD Data from SMPHOLD” on page 487

### Example 1: Report for Internal HOLDDATA

Figure 56 is an example of a RECEIVE Exception SYSMOD Summary report associated with the RECEIVE Summary report in Figure 59 on page 491. In that example, the ++HOLD statements are contained in the SYSMOD being held.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

RECEIVE ++HOLD/++RELEASE SUMMARY REPORT

NOTE: SMD NF - SYSMOD NOT RELEASED - NOT FOUND IN GLOBAL ZONE
      RSN NF - SYSMOD NOT RELEASED - NOT HELD FOR THIS REASONID
      INT HLD - SYSMOD NOT RELEASED - CANNOT RELEASE INTERNAL SYS HOLD

SYSMOD TYPE STATUS REASON FMID ++HOLD MCS STATEMENTS
JXY2102 SYS HELD FULLGEN JXY2102 ++HOLD(JXY2102) SYS FMID(JXY2102)
                                     REASON(FULLGEN)
                                     COMMENT(A FULL SYSGEN IS
                                     REQUIRED).
UZ00004 SYS HELD UCLIN JXY2102 ++HOLD(UZ00004) SYS FMID(JXY2102)
                                     REASON(UCLIN)
                                     COMMENT(THIS PTF REQUIRES UCLIN).
```

Figure 56. RECEIVE Exception SYSMOD Data Report: Sample Report for Internal HOLDDATA

**Example 2: Report for ++HOLD Data from SMPHOLD**

Figure 57 is an example of the report produced as a result of receiving the following ++HOLD statements from the SMPHOLD data set:

```

++HOLD(UZ00001) SYS FMID(JXY2102)
                REASON(UCLIN) COMMENT
                (UCLIN.
                DEL MOD(MODA) LMOD(IEANOC01).
                ENDUCL.).

++HOLD(UZ00002) ERR FMID(JXZ2102)
                REASON(AZ00004) DATE(90001).

++HOLD(AZ99991) USER FMID(JXY2102)
                REASON(EC) DATE(90001)
                COMMENT(NEEDS EC 123456).

++HOLD(MY00001) ERR FMID(HUSR001)
                REASON(MYAPAR1).

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

## RECEIVE ++HOLD/++RELEASE SUMMARY REPORT

NOTE: SMD NF - SYSMOD NOT RELEASED - NOT FOUND IN GLOBAL ZONE  
 RSN NF - SYSMOD NOT RELEASED - NOT HELD FOR THIS REASONID  
 INT HLD - SYSMOD NOT RELEASED - CANNOT RELEASE INTERNAL SYS HOLD

SYSMOD	TYPE	STATUS	REASON	FMID	++HOLD MCS STATEMENTS
AZ99991	USER	HELD	EC	JXY2102	++HOLD(AZ99991) USER FMID(JXY2102) REASON(EC) DATE(90001) COMMENT(NEEDS EC 123456).
UZ00001	SYS	HELD	UCLIN	JXY2102	++HOLD(UZ00001) SYS FMID(JXY2102) REASON(UCLIN) COMMENT (UCLIN. DEL MOD(MODA) LMOD(IEANOC01). ENDUCL.).
UZ00002	ERR	HELD	AZ00004	JXZ2102	++HOLD(UZ00002) ERR FMID(JXZ2102) REASON(AZ00004) DATE(90001).
MY00001	ERR	N/A	MYAPAR1	HUSR001	

Figure 57. RECEIVE Exception SYSMOD Data Report: Sample Report for External HOLDDATA

## RECEIVE Summary Report

This report is produced at the completion of RECEIVE processing to summarize the processing that occurred for SYSMODs and other data in SMPPTFIN. It is arranged by SYSMOD ID.

If **SOURCEID** was specified on the RECEIVE command, FOR SOURCEID=*source-id* appears on the report heading.

## Format and Explanation of Data

PAGE <i>nnnn</i> - NOW SET TO <i>zzzzzz</i> ZONE <i>nnnnnnn</i> DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT					
RECEIVE SUMMARY REPORT					
SYSMOD	STATUS	TYPE	SOURCEID	STATUS	FIELD COMMENTS
<i>aaaaaa</i>	<i>bbbbbbbbbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeeeeeeeeeeeeeeee</i>	
<i>aaaaaa</i>	<i>bbbbbbbbbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeeeeeeeeeeeeeeee</i>	
<i>aaaaaa</i>	<i>bbbbbbbbbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeeeeeeeeeeeeeeee</i>	
<i>aaaaaa</i>	<i>bbbbbbbbbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeeeeeeeeeeeeeeee</i>	
<i>aaaaaa</i>	<i>bbbbbbbbbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeeeeeeeeeeeeeeee</i>	
<i>aaaaaa</i>	<i>bbbbbbbbbbb</i>	<i>ccccccc</i>	<i>ddddddd</i>	<i>eeeeeeeeeeeeeeeeeeee</i>	

Figure 58. RECEIVE Summary Report: Standard Format

These are the fields in the report:

**SYSMOD**

identifies the SYSMOD that was processed.

**STATUS**

describes the outcome of the processing SMP/E did for the SYSMOD. It can be one of the following:

**ASSIGNED**

The source ID was assigned to the SYSMOD by the ++ASSIGN statement.

**NOT ASSIGNED**

The source ID specified by the ++ASSIGN statement was not assigned to the SYSMOD, because the SYSMOD was not already received.

**NOT RECEIVED**

An error occurred during SYSMOD processing.

**RECEIVED**

The SYSMOD was successfully received.

**RE-RECEIVED**

A reworked version of the SYSMOD was successfully received again.

**TYPE**

is the SYSMOD type: APAR, FUNCTION, PTF, or USERMOD.

**SOURCEID**

is the source ID specified on the ++ASSIGN statement associated with this SYSMOD.

**STATUS FIELD COMMENTS**

is additional information about the SYSMOD. It can be one of the following:

**\*NO SYSMODS PROCESSED\***

No SYSMODs were received.

**\*SYSMODS WITH SPECIFIED FMID(S) NOT FOUND IN SMPPTFIN\***

Although **RECEIVE FORFMID**(xxxxxxx) was specified, no SYSMODs were received, because there were no SYSMODs in the SMPPTFIN data set with an FMID matching the FMIDs specified on the FORFMID operand.

**ALREADY RECEIVED**

The SYSMOD was already in the CSI and PTS data sets and, therefore, was not received.

**CONSTRUCTION ERROR**

The SYSMOD was not constructed correctly. See SMPOUT for messages describing this error.

**I/O ERROR DURING PROCESSING**

A severe error occurred on one of the SMP/E data sets during SYSMOD processing. See SMPOUT for messages describing this error.

**NO APPLICABLE ++VER**

The SYSMOD did not contain a ++VER statement with SREL and FMID values that matched any SREL and FMID values in the global zone definition.

**RELFILES NOT PROCESSED**

An error occurred while SMP/E was processing the RELFILEs associated with this SYSMOD. See SMPOUT for messages describing this error.

**RELFILE PROCESSING ERROR**

An error occurred while SMP/E was processing the RELFILEs associated with this SYSMOD. See SMPOUT for messages describing this error.

**REWORK LEVEL IS NOT GREATER THAN THE VERSION ALREADY RECEIVED**

The SYSMOD was not received again, because the REWORK level on its header MCS was not higher than the REWORK level for the previously received version of that SYSMOD.

**SELECTED SYSMOD NOT FOUND ON SMPPTFIN**

The SYSMOD was specified on the SELECT operand but was not in the input data set.

**SMPTLIB LOADED**

SMP/E has successfully loaded the RELFILE data sets for this SYSMOD.

## STOPPED BY EXIT ROUTINE

The RECEIVE installation exit routine passed SMP/E a return code indicating that the SYSMOD should not be received.

## SYNTAX ERROR

SMP/E found a syntax error in at least one of the SYSMOD's MCSs. See SMPOUT for messages that describe this error.

## SYSMOD EXCLUDED

The SYSMOD was specified on the EXCLUDE operand of the RECEIVE command.

## SYSMOD NOT IN RECEIVE STATUS

The SYSMOD was specified on an ++ASSIGN statement but was not in the SMPPTS data set; therefore, the source ID could not be assigned.

## Examples

The following sample reports are provided:

- "Example 1: Successful RECEIVE"
- "Example 2: Unsuccessful RECEIVE" on page 491

### Example 1: Successful RECEIVE

This example shows a RECEIVE command from SMPPTFIN and the resulting RECEIVE Summary report formatted output:

```
SET      BDY(GLOBAL).      /* Set to global zone.      */
RECEIVE  SOURCEID(IP09006). /* Receive data.           */

++ASSIGN SOURCEID(PUT9304) TO(UZ00011,UZ00012).
++ASSIGN SOURCEID(XAU3380) TO(UZ00011,UZ00014).
++PTF(UZ00011).
++VER(Z038) FMID(JXY2101).
++MOD(A) DISTLIB(DN554).
A
++PTF(UZ00012).
++VER(Z038) FMID(F123456).
++MOD(C) DISTLIB(DN554).
C
++ASSIGN SOURCEID(PUT9305) TO(UZ00013,UZ00014).
++PTF(UZ00013).
++VER(Z038) FMID(JXY2101).
++IF FMID(JXY2102) THEN REQ(UZ00014).
++MOD(D) DISTLIB(DN554).
D
++PTF(UZ00014).
++VER(Z038) FMID(JXY2101) PRE(UZ00011).
++MOD(A) DISTLIB(A0S12).
A
++MOD(B) DISTLIB(A0S12).
B
++ASSIGN SOURCEID(PUT9304) TO(UZ70249).
++ASSIGN SOURCEID(PUT9305) TO(UZ60179).
/*
```

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
```

RECEIVE SUMMARY REPORT FOR SOURCEID = IPO9006

SYSMOD	STATUS	TYPE	SOURCEID	STATUS FIELD	COMMENTS
UZ00011	RECEIVED	PTF			
	ASSIGNED		PUT9304		
	ASSIGNED		XAU3380		
UZ00012	RECEIVED	PTF			
	ASSIGNED		PUT9304		
UZ00013	RECEIVED	PTF			
	ASSIGNED		PUT9305		
UZ00014	RECEIVED	PTF			
	ASSIGNED		PUT9305		
	ASSIGNED		XAU3380		
UZ60179	ASSIGNED		PUT9305		
UZ70249	NOT ASSIGNED		PUT9304		SYSMOD NOT IN RECEIVE STATUS

Figure 59. RECEIVE Summary Report: Sample Report for Successful RECEIVE Processing

### Example 2: Unsuccessful RECEIVE

Figure 60 shows a formatted report output of an unsuccessful RECEIVE run.

Assume the following command was entered:

```
SET      BDY(GLOBAL)      /* Set to global zone.      */.
RECEIVE S(UZ00001,        /* Receive two SYSMODs.     */.
        UZ00002)          /*                             */.
        SOURCEID(PUT9301) /* Assign this SOURCEID.    */.
```

Also assume the SMPPTFIN input contained the following:

```
++FUNCTION(JXY2102).
++VER(Z038).
++HOLD(JXY2102)
  SYSTEM
  FMID(JXY2102)
  REASON(FULLGEN)
  COMMENT(A FULL SYSGEN MUST BE PERFORMED TO
          INSTALL THIS FUNCTION)

++MOD(MOD0003) DISTLIB(DLIB) TXLIB(TXLIB).
++PTF(UZ00001).
++VER(Z038) FMID(JXY2102).
++MOD(MOD0003) DISTLIB(DLIB) TXLIB(TXLIB).
```

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
```

RECEIVE SUMMARY REPORT FOR SOURCEID=PUT9301

SYSMOD	STATUS	TYPE	SOURCEID	STATUS FIELD	COMMENTS
UZ00001	RECEIVED	PTF			
UZ00002	NOT RECEIVED				SELECTED SYSMOD NOT FOUND ON SMPPTFIN

Figure 60. RECEIVE Summary Report: Sample Report with Failing SYSMOD

## REJECT Summary Report

This report is produced at the completion of REJECT processing to summarize the processing that occurred for SYSMODs and other data.

### Format and Explanation of Data

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

REJECT SUMMARY REPORT FOR mmmmmm MODE ssssss

ZONES CHECKED NOTE: * - zoneset name # - zone name in a zoneset

DLIB - xddddddd xddddddd xddddddd xddddddd
      xdxxxxxx xddxxxxxx xddxxxxxx xddxxxxxx

TARGET - xttttttt xttttttt xttttttt xttttttt
         xttttttt xttttttt xttttttt xttttttt

NOT REJECTED REPORT

FMIDS NOT DELETED FROM THE GLOBAL ZONE FMID SUBENTRY

ffffff fffffff fffffff fffffff fffffff fffffff
ffffff fffffff fffffff fffffff fffffff fffffff

SYSMODS NOT REJECTED

SYSMOD COMMENTS

sssssss ccccccccccccccccccccccccccccccccccccc
sssssss ccccccccccccccccccccccccccccccccccccc

SUCCESSFULLY REJECTED REPORT

FMIDS DELETED FROM THE GLOBAL ZONE FMID SUBENTRY

ffffff fffffff fffffff fffffff fffffff fffffff
ffffff fffffff fffffff fffffff fffffff fffffff

SYSMODS REJECTED

sssssss sssssss sssssss sssssss sssssss sssssss
sssssss sssssss sssssss sssssss sssssss sssssss
```

Figure 61. REJECT Summary Report: Standard Format

These are the fields at the **beginning** of the report:

*mmmmmm*

is the mode of REJECT processing that was done: MASS, SELECT, PURGE, or NOFMID.

*ssssss*

is additional information appearing on continuation pages of the report to identify the part of the report that is being continued. It appears only when the report spans more than one page. This additional information may be one of the following:



- NOT REJECTED REPORT
- SYSMODS NOT REJECTED
- SUCCESSFULLY REJECTED REPORT

**DLIB**

lists the distribution zones that were checked during REJECT processing.

For mass or select mode, DLIB shows every distribution zone that was defined by a zone index in the GLOBALZONE entry, unless the distribution zone was specified on the EXCLUDEZONE operand.

For PURGE mode, DLIB shows the distribution zones and ZONESETs that were specified, as well as the distribution zones contained in any specified ZONESETs. Each ZONESET name is preceded with an asterisk (\*), and each distribution zone in the ZONESET is preceded with a pound sign (#).

If no distribution zones were checked, NONE appears instead of a zone name.

**TARGET**

lists the target zones that were checked during REJECT processing.

For mass or select mode, TARGET shows every target zone that was defined by a zone index in the GLOBALZONE entry, unless the target zone was specified on the EXCLUDEZONE operand.

For PURGE mode, TARGET shows the target zones and ZONESETs that were specified, as well as the target zones contained in any specified ZONESETs. Each ZONESET name is preceded with an asterisk (\*), and each target zone in the ZONESET is preceded with a pound sign (#).

If there are no entries for this heading, NONE appears instead of a zone name.

These are the fields in the **NOT REJECTED** section of the report:

**FMIDS NOT DELETED FROM THE GLOBAL ZONE FMID SUBENTRY**

lists each FMID specified on the DELETEDFMID operand that did not exist in the GLOBALZONE entry. FMIDs are listed only for NOFMID mode processing.

If there are no entries for this heading, NONE appears instead of an FMID.

**SYSMOD**

lists each SYSMOD that was a candidate to be rejected but then became ineligible. SYSMODs are only listed for select or PURGE mode processing.

If some SYSMODs were successfully rejected and none failed, NONE appears instead of a SYSMOD ID.

**COMMENTS**

explains why the SYSMOD was not rejected:

**SYSMOD ACCEPTED IN ZONE *dlibzone***

The SYSMOD was accepted in the specified distribution zone, and **BYPASS(ACCEPTCHECK)** was not specified with the SELECT operand.

**SYSMOD APPLIED IN ZONE *tgtzone***

The SYSMOD was applied in the specified target zone, and **BYPASS(APPLYCHECK)** was not specified with the SELECT operand.

### **SYSMOD NOT ACCEPTED IN ZONE** *dlibzone*

The SYSMOD was accepted in one or more of the distribution zones indicated on the PURGE operand. However, it was not accepted in the specified zone, where it was applicable.

### **SYSMOD NOT APPLIED IN ZONE** *tgtzone*

The SYSMOD was accepted in one or more of the distribution zones indicated on the PURGE operand. However, it was not applied in this target zone, which was specified on the TARGETZONE operand and in which the SYSMOD was applicable.

### **SYSMOD NOT FOUND IN THE GLOBAL ZONE**

A SYSMOD specified on the SELECT operand was not in the global zone. It might not have been received, or it might have already been rejected.

### **THERE ARE NO SYSMODS TO REJECT**

None of the SYSMODs in the global zone met the criteria specified on the REJECT command. No SYSMODs were rejected. No SYSMODs are listed in the NOT REJECTED section of the report, and NONE appears in the SUCCESSFULLY REJECTED section.

These are the fields in the **SUCCESSFULLY REJECTED** section of the report:

#### **FMIDS DELETED FROM THE GLOBAL ZONE FMID SUBENTRY**

lists each FMID that was deleted from the GLOBALZONE entry.

For NOFMID mode processing, it shows each FMID that was specified on the DELETEDFMID operand and that was successfully deleted.

For mass-mode and select-mode processing, it shows FMIDs of function SYSMODs that were not applied or accepted anywhere and were successfully deleted.

If there are no entries for this heading, NONE appears rather than an FMID.

#### **SYSMODS REJECTED**

lists each SYSMOD that was rejected.

If there are no entries for this heading, NONE appears rather than a SYSMOD ID.

## Examples

The following sample reports are provided:

- “Example 1: REJECT Summary Report for PURGE-Mode Processing” on page 495
- “Example 2: REJECT Summary Report for NOFMID-Mode Processing” on page 496
- “Example 3: REJECT Summary for Mass-Mode Processing” on page 497

### Example 1: REJECT Summary Report for PURGE-Mode Processing

Suppose you defined a ZONESET named MVSSET containing both target and distribution zones. You want to reject all SYSMODs that have been installed in the zones defined in MVSSET; therefore, you entered these commands:

```
SET BDY(GLOBAL).  
REJECT PURGE(MVSSET) TZONE(MVSSET).
```

Figure 62 is an example of a REJECT Summary report that you might see after running these commands.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT  
  
REJECT SUMMARY REPORT FOR PURGE MODE  
  
ZONES CHECKED NOTE: * - zoneset name # - zone name in a zoneset  
  
DLIB - *MVSSET #DMVA  
TARGET - *MVSSET #TMV1 #TMV2  
  
NOT REJECTED REPORT  
  
FMIDS NOT DELETED FROM THE GLOBAL ZONE FMID SUBENTRY  
  
(NONE)  
  
SYSMODS NOT REJECTED  
  
SYSMOD COMMENTS  
  
UZ01434 SYSMOD NOT APPLIED IN ZONE TMV2  
  
SUCCESSFULLY REJECTED REPORT  
  
FMIDS DELETED FROM THE GLOBAL ZONE FMID SUBENTRY  
  
(NONE)  
  
SYSMODS REJECTED  
  
UZ01245  
UZ01345  
UZ01723  
UZ01803
```

Figure 62. REJECT Summary Report: Sample Report for PURGE-Mode Processing

**Example 2: REJECT Summary Report for NOFMID-Mode Processing**

Assume you had planned to install function HMX1101, but then decided not to install it. You want to delete the FMID from the global zone and reject any service and HOLDDATA for that deleted FMID; therefore, you entered these commands:

**SET BDY(GLOBAL).**  
**REJECT DELETEFMID(HMX1101) NOFMID.**

Figure 63 is an example of a REJECT Summary report that you might see after running these commands.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

      REJECT SUMMARY REPORT FOR NOFMID MODE

ZONES CHECKED NOTE: * - zoneset name # - zone name in a zoneset

DLIB   - (NONE)
TARGET - (NONE)

NOT REJECTED REPORT

FMIDS NOT DELETED FROM THE GLOBAL ZONE FMID SUBENTRY

(NONE)

SYSMODS NOT REJECTED

SYSMOD COMMENTS

(NONE)

SUCCESSFULLY REJECTED REPORT

FMIDS DELETED FROM THE GLOBAL ZONE FMID SUBENTRY

HMX1101

SYSMODS REJECTED

UZ01245
UR02512
UR02522
```

Figure 63. REJECT Summary Report: Sample Report for NOFMID-Mode Processing

**Example 3: REJECT Summary for Mass-Mode Processing**

Assume you want to reject all SYSMODs that have not been applied or accepted; so you entered these commands:

```
SET BDY(GLOBAL).  
REJECT.
```

Figure 64 is an example of a REJECT Summary report that you might see after running these commands.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT  
  
REJECT SUMMARY REPORT FOR MASS MODE  
  
ZONES CHECKED NOTE: * - zoneset name # - zone name in a zoneset  
  
DLIB - DMVA  
TARGET - TMV1 TMV2  
  
NOT REJECTED REPORT  
  
FMIDS NOT DELETED FROM THE GLOBAL ZONE FMID SUBENTRY  
  
(NONE)  
  
SYSMODS NOT REJECTED  
  
SYSMOD COMMENTS  
  
(NONE)  
  
SUCCESSFULLY REJECTED REPORT  
  
FMIDS DELETED FROM THE GLOBAL ZONE FMID SUBENTRY  
  
(NONE)  
  
SYSMODS REJECTED  
  
UZ01245 UZ02512  
UZ01629 UZ02522  
UZ01845  
UZ01852  
UZ01924
```

Figure 64. REJECT Summary Report: Sample Report for Mass-Mode Processing

## SOURCEID Report

This report is produced for REPORT SOURCEID processing to summarize the source IDs found in the specified zones. A separate report is written for each zone specified on the REPORT SOURCEID command.

The format of the report depends on whether the **SYSMODIDS** operand was specified on the REPORT SOURCEID command.

## Format and Explanation of Data

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

SOURCEID REPORT FOR aaaaaa ZONE bbbbbbb

NOTE: * - INDICATES THE SYSMOD HAS MULTIPLE SOURCEIDS

SOURCEID      _____ SYSMOD NAMES _____
cccccccc      ddddddd ddddddd ddddddd ddddddd ddddddd
               ddddddd ddddddd ddddddd ddddddd
cccccccc      ddddddd ddddddd ddddddd
    
```

Figure 65. SOURCEID Report: Standard Format (SYSMODIDS Operand Specified)

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

SOURCEID REPORT FOR aaaaaa ZONE bbbbbbb

_____ SOURCEIDS _____

cccccccc ccccccc ccccccc ccccccc ccccccc ccccccc
cccccccc ccccccc ccccccc
    
```

Figure 66. SOURCEID Report: Standard Format (SYSMODIDS Operand Not Specified)

These are the fields in the report:

**aaaaaa**  
is the type of the zone being reported on.

**bbbbbb**  
is the name of the zone being reported on.

**SOURCEID(S)**  
is a source ID assigned to a SYSMOD in the indicated zone.

If none of the SYSMODs in the zone have been assigned a source ID, **\*\*\*NONE** appears in this field.

**SYSMOD NAMES**  
lists the ID of each SYSMOD to which the indicated source ID is assigned.

The SYSMOD ID only appears when **SYSMODIDS** was specified on the REPORT SOURCEID command.

## Examples

The following sample reports are provided:

- "Example 1: SYSMODIDS Operand Specified"
- "Example 2: SYSMODIDS Operand Not Specified"

### Example 1: SYSMODIDS Operand Specified

Assume you entered these commands:

```
SET BDY(GLOBAL).
REPORT SOURCEID
      ZONES(TGT1)
      SYSMODIDS.
```

Figure 67 is an example of the report SMP/E writes:

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

SOURCEID REPORT FOR TARGET ZONE TGT1

NOTE: * - INDICATES THE SYSMOD HAS MULTIPLE SOURCEIDS

SOURCEID _____ SYSMOD NAMES _____
PUT9303  UZ00023* UZ00024  UZ00035  UZ00037  UZ00039
         UZ00052  UZ00073  UZ00076  UZ00077
PUT9304  UZ00015  UZ00023* UZ00044
```

Figure 67. SOURCEID Report: Sample Report (SYSMODIDS Operand Specified)

### Example 2: SYSMODIDS Operand Not Specified

Assume you entered these commands:

```
SET BDY(GLOBAL).
REPORT SOURCEID
      ZONES(TGT2).
```

Figure 68 is an example of the report SMP/E writes:

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

SOURCEID REPORT FOR TARGET ZONE TGT2

_____ SOURCEIDS _____
PUT9206  PUT9207  PUT9208  PUT9301  PUT9302  PUT9303
PUT9304
```

Figure 68. SOURCEID Report: Sample Report (SYSMODIDS Operand Not Specified)

## SYSMOD Comparison Report

This report is produced for REPORT SYSMODS processing to summarize the SYSMODs found in the input zone, but not found in the comparison zone. If no such SYSMODs are found in the input zone, the SYSMOD Comparison report states THERE WERE NO SYSMODS TO REPORT.

### Format and Explanation of Data

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
SYSMOD COMPARISON REPORT FOR ztype ZONE zone1 AND ztype ZONE zone2
MATCHING SREL(S) - aaaa, bbbb, ...
FMID__ SYSMOD_ TYPE___ APPLICABLE RECEIVED SOURCEIDS
cccccc ddddddd eeeeeeee ffffffff ggg          hhhhhhhh
          ddddddd eeeeeeee ffffffff ggg          hhhhhhhh
cccccc ddddddd eeeeeeee ffffffff ggg          hhhhhhhh
          ddddddd eeeeeeee ffffffff ggg          hhhhhhhh
          ddddddd eeeeeeee ffffffff ggg          hhhhhhhh
```

Figure 69. SYSMOD Comparison Report: Standard Format

These are the fields in the report:

**ztype**

is the zone type being reported on. It may be either TARGET or DLIB.

**zone1**

is the name of the input zone.

**zone2**

is the name of the zone being compared against the input zone (called the *comparison zone*).

**MATCHING SREL(S)**

are the SRELS that match in the zones being compared. These are the SRELS defined in the zone definition entries.

**FMID**

is the FMID of the SYSMOD being reported on (shown in the SYSMOD field).

**SYSMOD**

is the ID of a SYSMOD installed in the input zone, but not found in the comparison zone.

**Note:** This field does not show superseded-only SYSMODs, SYSMODs with the DELBY subentry, or SYSMODs in error, because these types of SYSMODs are not individually installed or are not completely installed.

**TYPE**

is the SYSMOD type of the SYSMOD being reported on: APAR, FUNCTION, PTF, or USERMOD.



**APPLICABLE**

indicates whether the SYSMOD being reported on is applicable to the comparison zone. The value in this field can be one of the following:

**YES** The SYSMOD is applicable to the comparison zone.

**NO** The SYSMOD is not applicable to the comparison zone.

**REINSTALL**

The SYSMOD is a function that is installed in the comparison zone. However, the input zone contains a service update of that function that is applicable to the comparison zone. Therefore, you may want to reinstall the service-updated function in the comparison zone.

**UNKNOWN**

SMP/E cannot determine whether the SYSMOD is applicable to the comparison zone. This can happen if the input zone supports at least one SREL that is not supported by the comparison zone, and there is no entry for the SYSMOD or its FMID in the global zone.

To determine whether the SYSMOD is applicable to the comparison zone, you need to check the program directory or installation manual for the FMID to find out its SREL. If that SREL is supported by the comparison zone, the SYSMOD may be applicable. For more details, see the description of REPORT SYSMODS processing on page 325.

If the SYSMOD is applicable, receive it again, and change the SMP/PUNCH output so that the SYSMOD is no longer commented out. If you could not determine whether the SYSMOD is applicable, you can still use this approach. Messages issued during APPLY CHECK or ACCEPT CHECK processing indicate whether the SYSMOD is applicable.

**RECEIVED**

indicates whether the SYSMOD being reported on has been received and is available in the global zone. Either YES or NO may appear in this field.

**SOURCEIDS**

is a list of the source IDs associated with the SYSMOD in the input zone. If there are no source IDs associated with the SYSMOD, this field is blank.

**Note:** If an applicable SYSMOD is not received, the source IDs may help you determine where to obtain the SYSMOD.

## Example: SYSMOD Comparison Report

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
SYSMOD COMPARISON REPORT FOR TARGET ZONE MVSESA1 AND TARGET ZONE MVSESA2
MATCHING SREL(S) - Z038
```

FMID__	SYSMOD_	TYPE__	APPLICABLE	RECEIVED	SOURCEIDS
HAA1202	HAA1202	FUNCTION	YES	YES	
	UZ00011	PTF	YES	NO	PUT9306
	UZ00012	PTF	YES	YES	PUT9307
	AZ00013	APAR	YES	NO	RETAIN
	TZ00001	USERMOD	YES	YES	
HBB1202	UZ00002	PTF	YES	YES	PUT9307
					PD00001
JBB1222	UZ00021	PTF	NO	NO	PUT9306
	UZ00022	PTF	NO	YES	PUT9307

Figure 70. SYSMOD Comparison Report: Sample Report

# SYSMOD Regression Report

This report is produced at the completion of APPLY and ACCEPT processing to summarize which SYSMODs were regressed. Regression occurs when SMP/E installs an element from a SYSMOD that did not express a proper PRE or SUP relationship with the RMID and UMID values in the element entry. Regression can occur only when BYPASS(ID) is used to ignore such errors. It does not occur when a new function is being installed, because elements from a function are selected on the basis of functional superiority. SMP/E assumes that service (PTFs and USERMODS) installed on the functionally inferior elements provides ++IF conditional requisite data to ensure that the PTF or USERMOD is at the proper service level.

If no regressions are detected, this report is generally not produced. In certain cases, however, SMP/E detects a regression that is later resolved by other SYSMODs being processed by the same APPLY or ACCEPT command. In this case, the regression report is produced but contains just one message: NO SYSMODS REGRESSED.

## Format and Explanation of Data

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
SYSMOD REGRESSION REPORT FOR xxxxxxxx yyyy PROCESSING
NOTE: '*' INDICATES THAT THE REGRESSED MODID IS SUPD BY ANOTHER SYSMOD BEING PROCESSED. THE REGRESSION MAY BE RESOLVED.
      '- ' INDICATES THE CURRENT RMID IS A HIGHER LEVEL SYSMOD THAN THE REGRESSING SYSMOD
REGRESSING  REGRESSED  COMMON  ELEMENTS  CURRENT  OTHER POTENTIALLY
SYSMOD      SYSMOD      TYPE    NAME      RMID     REGRESSED SYSMODS
-----
aaaaaaa    bbbbbbb    ccccccc dddddddd eeeeeee  ffffffff ffffffff ffffffff
          bbbbbbb    ccccccc dddddddd eeeeeee  ffffffff ffffffff ffffffff
aaaaaaa    bbbbbbb    ccccccc dddddddd eeeeeee  ffffffff ffffffff ffffffff

```

Figure 71. SYSMOD Regression Report: Standard Format

These are the fields in the report:

xxxxxxx  
is the SMP/E command being processed: APPLY or ACCEPT.

yyyyy  
is CHECK if CHECK was specified on APPLY or ACCEPT. Otherwise, this field is blank.

**REGRESSING SYSMOD**  
identifies the SYSMOD that caused the listed elements to be regressed.

**REGRESSED SYSMOD**  
is a list of SYSMODs that had previously changed the elements listed in the COMMON ELEMENTS fields. These changes may have been overlaid.

**COMMON ELEMENTS TYPE and NAME**  
lists the elements changed by the regressing SYSMOD.

**CURRENT RMID**

is the RMID for the highest level SYSMOD replacing this element in this SMP/E run. If the element is not being replaced by the current SMP/E run, or is being deleted, this column may be blank.

**OTHER POTENTIALLY REGRESSED SYSMODS**

is a list of SYSMODs that were superseded by the regressed SYSMOD, but were not superseded by the regressing SYSMOD. This list may include the SYSMOD IDs of APARs that were fixed (superseded) by the regressed SYSMOD, but were not included in the regressing SYSMOD.

**Example: APPLY SYSMOD Regression Report**

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
SYSMOD REGRESSION REPORT FOR APPLY CHECK PROCESSING

NOTE: '*' INDICATES THAT THE REGRESSED MODID IS SUPD BY ANOTHER SYSMOD BEING PROCESSED. THE REGRESSION MAY BE RESOLVED.
      '- ' INDICATES THE CURRENT RMID IS A HIGHER LEVEL SYSMOD THAN THE REGRESSING SYSMOD
    
```

REGRESSING SYSMOD	REGRESSED SYSMOD	COMMON TYPE	ELEMENTS NAME	CURRENT RMID	OTHER POTENTIALLY REGRESSED SYSMODS
UZ00099	UZ00001	MODULE	HMAB0123		AZ00050 AZ00051 AZ00052
	UZ00002	MACRO MODULE	HMAMAC01 HMAB0456		AZ00055
	UZ00003	MODULE MODULE MODULE	HMAB0790 HMAB0012 HMAB0345		AZ00056 AZ00057
UZ00111	UZ00004	MODULE MODULE MODULE MODULE	HMAB0678 HMAB0987 HMAB0124 HMAB0135		AZ00058 AZ00059 AZ00060
UZ01000	UZ00100	MODULE MODULE SOURCE SOURCE SOURCE SOURCE	MOD002 MOD003 MOD002 MOD003 SRC002 SRC003	UZ01000 UZ01000 UZ01000 UZ01000	UZ00002 UZ00003 UZ00004 UZ00005 UZ00006 UZ00007 UZ00012 UZ00013 UZ00014 UZ00015 UZ00016 UZ00017

Figure 72. SYSMOD Regression Report: Sample Report for APPLY

## SYSMOD Status Report

This report is produced at the completion of APPLY, ACCEPT, and RESTORE processing to summarize the processing that occurred for every eligible SYSMOD. The SYSMODs are listed in alphanumeric order.

### Format and Explanation of Data

```

PAGE nnnn - NOW SET TO TARGET ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

SYSMOD STATUS REPORT FOR xxxxxxxx PROCESSING SYSMODS yyyyyyyy - nnnn

NOTE: '-' INDICATES THE REQUISITE SYSMOD OR HOLD CONDITION IS NOT SATISFIED
      '* ' INDICATES THE NON SATISFIED REQUISITE SYSMOD OR HOLD CONDITION IS BYPASSED
      '# ' INDICATES THE SUPERSEDING SYSMOD WAS NOT PROCESSED

SYSMOD STATUS TYPE FMID REQUISITE SYSMODS, SUPBY SYSMODS, HOLD REASON-IDS, AND CAUSER SYSMODS

aaaaaa bbbbbb cccccc dddddd eeeee fffffff...
aaaaaa bbbbbb cccccc dddddd eeeee fffffff...
aaaaaa bbbbbb cccccc dddddd eeeee fffffff...
aaaaaa bbbbbb cccccc dddddd eeeee fffffff...
aaaaaa bbbbbb cccccc dddddd eeeee fffffff...
    
```

Figure 73. SYSMOD Status Report: Standard Format

These are the fields in the report:

**xxxxxxx**

is the SMP/E command being processed: APPLY, ACCEPT, or RESTORE.

**yyyyyyy**

indicates the type of processing that was done: APPLIED, ACCEPTED, or RESTORED.

**nnnn**

is the number of SYSMODs with a status of APPLIED, ACCEPTED, or RESTORED; that number depends on the command that was processed.

#### SYSMOD

identifies the SYSMOD that was processed.

#### STATUS

describes what happened to the SYSMOD. It can be one of the following:

##### APPLIED, ACCEPTED, or RESTORED

The SYSMOD was successfully processed.

##### DELETED

The SYSMOD was explicitly or implicitly deleted.

##### ERROR

SYSMOD processing stopped after some target libraries or SMP/E libraries were updated, but before the SYSMOD was completely processed. A SYSMOD is completely processed when all its elements have been processed and all its requisites have been completely processed. See SMPDUMP to determine the cause of the error.

**Note:** ERROR does not appear when the **CHECK** operand is specified on the command.

**EXCLUDED**

The SYSMOD was specified on the EXCLUDE operand.

**HELD** The SYSMOD was held because one or more of HOLD reason IDs were not resolved.

**INCMPLT** SYSMOD processing is incomplete because of some failure. No target libraries were updated.

**NOGO** The SYSMOD was not processed before any updates. This can happen when a related SYSMOD has an error. See SMPOUT to determine the cause of the error.

**NOGO(E)** SYSMOD processing stopped because a required SYSMOD was excluded.

**NOGO(H)** SYSMOD processing stopped because a required SYSMOD was held.

**SUPD** The SYSMOD is superseded by one or more SYSMODs being processed. The superseding SYSMODs are shown in the REQUISITE AND SUPBY SYSMODS field.

**Note:** Not all superseded SYSMODs are listed in the report. For example, SYSMODs that were not selected for processing and appear only in another SYSMOD's ++VER SUP operand are not listed.

**TYPE**

is the SYSMOD type: APAR, FUNCTION, PTF, or USERMOD. For superseded SYSMODs that have not been received, this field is blank.

**FMID**

is the function SYSMOD that owns the SYSMOD. For superseded SYSMODs, this field is blank.

**REQUISITE SYSMODS, SUPBY SYSMODS, HOLD REASON-IDs, AND CAUSER SYSMODS**

lists SYSMODs or reason IDs associated with the SYSMODs being installed. If a SYSMOD was not installed, these SYSMODs or reason IDs are preceded by a character indicating why (-, \*, or #). The list of SYSMODs or reason IDs is preceded by one of the following values, which indicate the type of SYSMOD or reason ID.

**CAUSER** SYSMODs whose failure led to the failure of the SYSMOD in the SYSMOD field. All the SYSMODs in the CAUSER field are in the Causer SYSMOD Summary report, along with a summary of the related error and, when feasible, a list of possible causes for the error. This holds true even when the SYSMOD in the SYSMOD field and the causer SYSMOD are the same.

**HOLDE** ERROR reason IDs for the SYSMOD.

**HOLDS** SYSTEM reason IDs for the SYSMOD.

**HOLDU** USER reason IDs for the SYSMOD.

- IFREQ** Conditional requisites for the SYSMOD, as defined by its associated ++IF statements or, if the SYSMOD is a function, defined by previously processed SYSMODs.
- PRE** Prerequisites for the SYSMOD.
- REQ** Requisites for the SYSMOD.
- SUPBY** SYSMODs that supersede the SYSMOD.

For HOLDE, HOLDS, HOLDU, IFREQ, PRE, and REQ:

- A dash (-) next to a listed SYSMOD means that SYSMOD has NOGO status and may not be available for processing.
- If an asterisk (\*) appears next to a listed SYSMOD, that SYSMOD has NOGO status, but the appropriate option was specified in the BYPASS operand list on the APPLY or ACCEPT commands. This means that even if the SYSMOD is not available for processing, the SYSMOD that has specified it as a requisite can be processed.

For SUPBY:

- If a pound sign (#) appears next to a listed SYSMOD, that SYSMOD has not been successfully processed. As long as at least one superseding SYSMOD has been successfully processed, the superseded SYSMOD is considered to be installed.

### Example: APPLY SYSMOD Status Report

```

PAGE nnnn - NOW SET TO TARGET ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

SYSMOD STATUS REPORT FOR APPLY PROCESSING      SYSMODS APPLIED - 349

NOTE: '-' INDICATES THE REQUISITE SYSMOD OR HOLD CONDITION IS NOT SATISFIED
      '*' INDICATES THE NON SATISFIED REQUISITE SYSMOD OR HOLD CONDITION IS BYPASSED
      '#' INDICATES THE SUPERSEDING SYSMOD WAS NOT PROCESSED

SYSMOD  STATUS  TYPE    FMID    REQUISITE SYSMODS, SUPBY SYSMODS, HOLD REASON-IDS, AND CAUSER SYSMODS
UY15425  NOGO      PTF     HAE1500  IFREQ  -UZ63106
          :                               CAUSER  UY15425   UZ65387
          :
UZ62368  HELD      PTF     JTC2412  PRE    UZ59092  UZ60917
          :                               HOLDE  -AZ71745 -AZ75533
          :                               CAUSER  UZ62368
          :
UZ65356  NOGO(H)   PTF     JTC2412  PRE    -UZ63740
          :                               CAUSER  UZ63419  UZ63420
          :
UZ65375  NOGO(H)   PTF     HHM1302  IFREQ  UZ61932  UZ61934  UZ62802  65385  UZ65386  -UZ65387
          :                               -UZ65388  -UZ65389
          :                               REQ    UZ65376  UZ65377  UZ65378  65379  UZ65380  UZ65381
          :                               UZ65382  UZ65383  UZ65384
          :                               CAUSER  UZ65387  UZ65388
          :
  
```

Figure 74. SYSMOD Status Report: Sample Report for APPLY

## UNLOAD Summary Report

This report is produced during UNLOAD processing to summarize which entries were found in the set-to zone and which entries were not.

### Format and Explanation of Data

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
UNLOAD SUMMARY FOR aaaaaaaaa
ENTRY-TYPE  ENTRY-NAME  STATUS
bbbbbbbb  ccccccc  ddddddddddddddd
bbbbbbbb  ccccccc  ddddddddddddddd
bbbbbbbb  ccccccc  ddddddddddddddd
bbbbbbbb  ccccccc  ddddddddddddddd
```

Figure 75. UNLOAD Summary Report: Standard Format

These are the fields in the report:

**aaaaaaaa**

is the name of the zone containing the entries being unloaded.

**ENTRY-TYPE**

is the type of entry SMP/E looked for. If no specific entries of a given type were selected, that entry type is shown only once. If several specific entries of a given type were selected, that entry type is shown for each of the selected entries.

**ENTRY-NAME**

is the name of an entry specified on the UNLOAD command. If no specific entries were selected, this is blank.

**STATUS**

indicates whether any entries were found. The status may be one of the following:

**FOUND** The specified entry or entry type was found.

**NOT FOUND**

The specified entry or entry type was not found.

**EMPTY ZONE – NO ENTRIES FOUND**

This may appear for the **UNLOAD** command. No entries were found in the specified zone. This is the only line in the report. The entry type and entry name fields are blank.



## Example: UNLOAD Summary Report

Figure 76 shows the UNLOAD Summary report that may accompany the UNLOAD output for the following command:

**UNLOAD MOD MAC SRC CLIST(CLIST01,CLIST02,CLIST03).**

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
UNLOAD SUMMARY FOR MVSTGT1
ENTRY-TYPE  ENTRY-NAME  STATUS
CLIST       CLIST01     FOUND
CLIST       CLIST02     NOT FOUND
CLIST       CLIST03     FOUND
MAC         NOT FOUND
MOD         FOUND
SRC         FOUND
```

Figure 76. UNLOAD Summary Report: Sample Report

**ZONEEDIT Summary Report**

This report is produced during ZONEEDIT processing to show the DDDEF and UTILITY entry names that were changed.

**Format and Explanation of Data**

PAGE <i>nnnn</i> - NOW SET TO <i>zzzzz</i> ZONE <i>nnnnnn</i> DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT			
SUMMARY REPORT FOR ZONEEDIT			
ENTRY NAME	FIELD NAME	FROM	TO
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>ccccccc...</i>	<i>ddddddd...</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>ccccccc...</i>	<i>ddddddd...</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>ccccccc...</i>	<i>ddddddd...</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>ccccccc...</i>	<i>ddddddd...</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>ccccccc...</i>	<i>ddddddd...</i>

Figure 77. ZONEEDIT Summary Report: Standard Format

These are the fields in the report:

**ENTRY NAME**

is the name of the entry that was changed. The valid entry types are DDDEF, UTILITY, and XZENTRIES.

**Note:** XZENTRIES is not an actual entry type. It is used to change a zone name in all the cross-zone subentries in a specified zone.

**FIELD NAME**

is the subentry that was changed.

For a DDDEF entry, the subentry name can be any of the following:

- DATASET
- SYSOUT
- UNIT
- VOLUME
- WAIT

For a UTILITY entry, the subentry name can be any of the following:

- NAME
- PRINT

For XZENTRIES, the subentry name can be any of the following:

- XZLMOD in MOD entries
- XZMOD in LMOD entries
- TIEDTO in the TARGETZONE entry

**FROM**

is the old value of the subentry. It can be up to 44 characters long.

**TO**

is the new value of the subentry. It can be up to 44 characters long.

## Examples

The following sample reports are provided:

- “Example 1: ZONEEDIT Summary Report for DDDEF Entries”
- “Example 2: ZONEEDIT Summary Report for XZENTRIES”

### Example 1: ZONEEDIT Summary Report for DDDEF Entries

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

        SUMMARY REPORT FOR ZONEEDIT

ENTRY NAME   FIELD NAME   FROM      TO
AOS12        UNIT         3330      3350
AOS22        UNIT         3330      3350
LPALIB       UNIT         3330      3350
LINKLIB      UNIT         3330      3350
    
```

Figure 78. ZONEEDIT Summary Report: Sample Report for DDDEF Entries

### Example 2: ZONEEDIT Summary Report for XZENTRIES

Suppose you have used the LINK command to link-edit modules from zone CICS1 into load modules in zone MVS1. Later, because of new zone-naming conventions, you used the ZONERENAME command to rename zone CICS1 to CICSPRD. When the ZONERENAME processing was completed, you realized that zone MVS1 was the only zone connected to zone CICS1 by cross-zone subentries.

You now have to change the cross-zone subentries in zone MVS1 so that MVS1 is connected to the renamed zone, CICSPRD. The following ZONEEDIT commands make this change:

```

SET      BDY (MVS1)      /* Set to zone to edit.      */.
ZONEEDIT XZENTRIES      /* Edit cross-zone subentries.*/.
CHANGE   ZONEVALUE(CICS1 /* Change zone from old name */
          CICSPRD)      /* to new name.                */.
ENDZONEEDIT              /* End of ZONEEDIT.           */.
    
```

After SMP/E processes these commands, the ZONEEDIT Summary report that is produced is similar to the following:

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

NOW SET TO TARGET ZONE MVS1

        SUMMARY REPORT FOR ZONEEDIT

ENTRY NAME   FIELD NAME   FROM      TO
MVS1         TIEDTO      CICS1     CICSPRD
LMOD1        XZMOD       CICS1     CICSPRD
MOD1         XZLMOD      CICS1     CICSPRD
    
```

Figure 79. ZONEEDIT Summary Report: Sample Report for XZENTRIES

## ZONEMERGE Report

This report is produced during ZONEMERGE processing to show the entries that were copied. If an error occurs and command processing stops, you can use this report to determine how far the merge operation has been completed. This report is arranged by entry type and, within entry type, alphanumerically by entry name. If no entries were merged, the ZONEMERGE report states NO ENTRIES MERGED. NO APPLICABLE ENTRIES IN FROM ZONE.

## Format and Explanation of Data

PAGE <i>nnnn</i> - NOW SET TO <i>zzzzzz</i> ZONE <i>nnnnnnn</i> DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPRPT OUTPUT			
SMP/E ZONEMERGE REPORT FROM ZONE <i>xxxxxxx</i> TO ZONE <i>yyyyyyy</i>			
TYPE	NAME	ACTION	REASON
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>cccccccc</i>	<i>dddddddddddddddddddd</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>cccccccc</i>	<i>dddddddddddddddddddd</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>cccccccc</i>	<i>dddddddddddddddddddd</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>cccccccc</i>	<i>dddddddddddddddddddd</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>cccccccc</i>	<i>dddddddddddddddddddd</i>
<i>aaaaaaaa</i>	<i>bbbbbbbb</i>	<i>cccccccc</i>	<i>dddddddddddddddddddd</i>

Figure 80. ZONEMERGE Report: Standard Format

These are the fields in the report:

*xxxxxxx*

is the zone containing the entries to be copied, also called the *FROM zone*.

*yyyyyyy*

is the zone to which you are copying entries, also called the *TO zone*.

**TYPE**

is the entry type.

**NAME**

is the name of the entry.

**ACTION**

describes what SMP/E did with that entry. It may be one of the following:

**MERGED** The specified entry was in the FROM zone but not in the TO zone; so SMP/E added it to the TO zone.

**REPLACED**

The specified entry was in both the FROM zone and the TO zone. Because **REPLACE** was specified on ZONEMERGE, SMP/E replaced the entry in the TO zone with the entry in the FROM zone.

**NOT MERGED**

The specified entry was in both the FROM zone and the TO zone. Because **NOREPLACE** was specified on ZONEMERGE, SMP/E did not replace the entry in the TO zone.

**REASON**

is the reason the entry was not merged if the ACTION field shows NOT MERGED. The only value for this field is REPLACE NOT SPECIFIED.

## Examples

The following sample reports are provided:

- “Example 1: Merge to Null Zone”
- “Example 2: Merge to Existing Zone with REPLACE Operand” on page 514
- “Example 3: Merge to Existing Zone with NOREPLACE Operand” on page 515

### Example 1: Merge to Null Zone

Assume you are merging zone TGT1 into a null zone TGT2. Figure 81 is an example of the ZONEMERGE report when the TO zone is null.

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

      SMP/E ZONEMERGE REPORT FROM ZONE TGT1   TO ZONE TGT2

TYPE      NAME      ACTION      REASON

DDDEF     AMACLIB    MERGED
DDDEF     ASAMPLIB    MERGED
ASSEM     ASSEM01     MERGED
ASSEM     ASSEM02     MERGED
ASSEM     ASSEM03     MERGED
LMOD      LMOD01      MERGED
LMOD      LMOD02      MERGED
LMOD      LMOD03      MERGED
MACRO     MAC01       MERGED
MACRO     MAC02       MERGED
MACRO     MAC03       MERGED
MODULE    MOD01       MERGED
MODULE    MOD02       MERGED
MODULE    MOD03       MERGED
SOURCE    SRC01       MERGED
SOURCE    SRC02       MERGED
SOURCE    SRC03       MERGED
DLIB      AMACLIB     MERGED
DLIB      ASRCLIB     MERGED
SYSMOD    AZ10001     MERGED
SYSMOD    AZ10002     MERGED
SYSMOD    UZ00001     MERGED
SYSMOD    UZ00002     MERGED
SYSMOD    UZ00003     MERGED

```

Figure 81. ZONEMERGE Report: Sample Report for Merging to a Null Zone

**Example 2: Merge to Existing Zone with REPLACE Operand**

If zone TGT2 contained the following entries:

```
DDDEF    ASAMPLIB
ASSEM    ASSEM01
LMOD     LMOD02
LMOD     LMOD03
MACRO    MAC02
MODULE   MOD02
SOURCE   SRC03
```

and you specified the following commands:

```
SET      BDY(TGT2)      /* Set to target zone.    */.
ZMRG     (TGT1)         /* Merge TGT1            */.
          INTO(TGT2)    /* into TGT2,           */.
          REPLACE       /* replacing like entries.*/.
```

the ZONEMERGE report looks like Figure 82.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT
SMP/E ZONEMERGE REPORT FROM ZONE TGT1 TO ZONE TGT2
```

TYPE	NAME	ACTION	REASON
DDDEF	AMACLIB	MERGED	
DDDEF	ASAMPLIB	REPLACED	
ASSEM	ASSEM01	REPLACED	
ASSEM	ASSEM02	MERGED	
ASSEM	ASSEM03	MERGED	
LMOD	LMOD01	MERGED	
LMOD	LMOD02	REPLACED	
LMOD	LMOD03	REPLACED	
MACRO	MAC01	MERGED	
MACRO	MAC02	REPLACED	
MACRO	MAC03	MERGED	
MODULE	MOD01	MERGED	
MODULE	MOD02	REPLACED	
MODULE	MOD03	MERGED	
SOURCE	SRC01	MERGED	
SOURCE	SRC02	MERGED	
SOURCE	SRC03	REPLACED	
DLIB	AMACLIB	MERGED	
DLIB	ASRCLIB	MERGED	
SYSMOD	AZ10001	MERGED	
SYSMOD	AZ10002	MERGED	
SYSMOD	UZ00001	MERGED	
SYSMOD	UZ00002	MERGED	
SYSMOD	UZ00003	MERGED	

Figure 82. ZONEMERGE Report: Sample Report for REPLACE Processing

**Example 3: Merge to Existing Zone with NOREPLACE Operand**

Assuming the same two zones were merged and you specified the following commands:

```

SET      BDY(TGT2)          /* Set to target zone.    */.
ZMRG     (TGT1)             /* Merge TGT1             */.
          INTO(TGT2)        /* into TGT2.            */.
          NOREPLACE         /* Don't merge like entries.*/.

```

the ZONEMERGE report looks like Figure 83.

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPRPT OUTPUT

```

SMP/E ZONEMERGE REPORT FROM ZONE TGT1 TO ZONE TGT2			
TYPE	NAME	ACTION	REASON
DDDEF	AMACLIB	MERGED	
DDDEF	ASAMPLIB	NOT MERGED	REPLACE NOT SPECIFIED
ASSEM	ASSEM01	NOT MERGED	REPLACE NOT SPECIFIED
ASSEM	ASSEM02	MERGED	
ASSEM	ASSEM03	MERGED	
LMOD	LMOD01	MERGED	
LMOD	LMOD02	MERGED	
LMOD	LMOD03	MERGED	
MACRO	MAC01	MERGED	
MACRO	MAC02	NOT MERGED	REPLACE NOT SPECIFIED
MACRO	MAC03	MERGED	
MODULE	MOD01	MERGED	
MODULE	MOD02	NOT MERGED	REPLACE NOT SPECIFIED
MODULE	MOD03	MERGED	
SOURCE	SRC01	MERGED	
SOURCE	SRC02	MERGED	
SOURCE	SRC03	NOT MERGED	REPLACE NOT SPECIFIED
DLIB	AMACLIB	MERGED	
DLIB	ASRCLIB	MERGED	
SYSMOD	AZ10001	MERGED	
SYSMOD	AZ10002	MERGED	
SYSMOD	UZ00001	MERGED	
SYSMOD	UZ00002	MERGED	
SYSMOD	UZ00003	MERGED	

Figure 83. ZONEMERGE Report: Sample Report for NOREPLACE Processing





---

## Chapter 32. SMP/E Modification Control Statements

Each SYSMOD processed by SMP/E is composed of two distinct types of data: instructions to SMP/E identifying the elements in the SYSMOD and how to install them, and the actual element replacements or updates. The instructions to SMP/E consist of a series of control statements, called modification control statements (or MCSs). This chapter describes the various MCSs that are processed by SMP/E.

### Building SYSMODs (Packaging)

Building SYSMODs (“packaging”) includes combining the appropriate MCS statements with software elements to create one or more SYSMODs. Depending on the type of SYSMODs you are building and how you plan to distribute them, packaging can also involve putting the SYSMODs in the proper format on the distribution medium.

Although this book describes the syntax of SMP/E MCS statements, it does not contain all the information you need to use these statements for packaging SYSMODs.

- To package function SYSMODs and the associated service (PTF SYSMODs and APAR SYSMODs), you must use this book along with the *Standard Packaging Rules for MVS-Based Products* manual, which contains the rules, restrictions, and recommendations for packaging SYSMODs.
- To package USERMOD SYSMODs, use this book along with the *SMP/E R8.1 User's Guide*, which steps you through building a USERMOD and provides USERMOD examples that you might find helpful.

### Notes:

1. Each section describing an individual MCS has examples of SYSMODs containing that MCS. In the examples, the MCS being described is underlined. This is done only to make that MCS stand out; it does not imply that any special processing must be done to enter that data.
2. The examples of MCSs do not show the use of all the operands for each MCS. When you want to know how to use a particular operand for a specific MCS, first check the section describing that MCS. If the operand is not shown in an example there, check the index entry for the desired **operand** to see which other MCSs also contain that operand. Then check the examples under those MCSs. Examples of the use of an operand for one MCS can often illustrate its use for another MCS.

The following chart lists the MCSs and the page on which each is found.

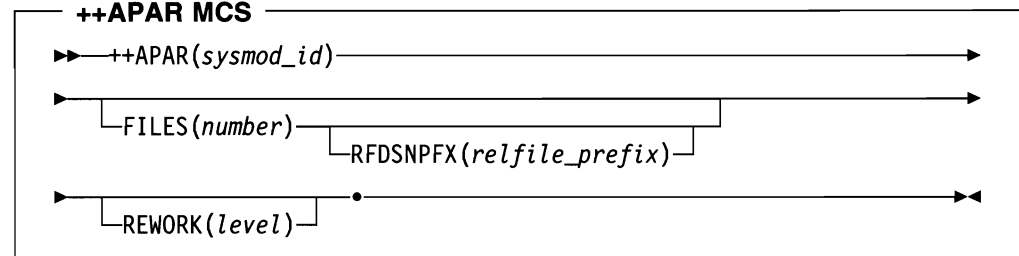
---

<b>MCS</b>	<b>Page</b>	<b>MCS</b>	<b>Page</b>
++APAR MCS	519	++MOD MCS	563
++ASSIGN MCS	521	++MOVE MCS	570
Data Element MCS	523	++NULL MCS	573
++DELETE MCS	529	++PTF MCS	574
++FUNCTION MCS	533	++RELEASE MCS	576
++HFS MCS	535	++RENAME MCS	580
++HOLD MCS	540	++SRC MCS	582
++IF MCS	546	++SRCUPD MCS	586
++JCLIN MCS	548	++USERMOD MCS	589
++MAC MCS	553	++VER MCS	591
++MACUPD MCS	559	++ZAP MCS	598

## ++APAR MCS

The ++APAR MCS identifies a service SYSMOD. This type of SYSMOD is a temporary corrective fix to the elements of target system and distribution libraries. All other MCSs for this SYSMOD follow this header MCS. For more information about packaging an APAR fix, see the *Standard Packaging Rules for MVS-Based Products* manual.

### Syntax



### Operands

#### FILES

specifies the number of relative files belonging to this APAR fix. It can be a decimal number from 1 to 9999. For information about packaging SYSMODs in relative files, see the *Standard Packaging Rules for MVS-Based Products* manual.

#### Notes:

1. Although SMP/E allows you to package APAR fixes in relative files, they are not generally packaged in this format.
2. If a packager uses a high-level qualifier on RELFILE data sets, the RFDSNPFX operand on the header MCS (not the RFPREFIX operand on the RECEIVE command) **must** be used to identify that high-level qualifier.

#### REWORK

specifies the level of this SYSMOD, which was reworked for minor changes. Up to eight numeric characters can be specified.

For SYSMODs supplied by IBM, the REWORK level is *yyyyddd*, where *yyyy* is the year the SYSMOD was reworked and *ddd* is the Julian date.

REWORK allows an updated SYSMOD to be automatically received again, as long as it is more recent than the version that has already been received. This takes the place of rejecting the SYSMOD and receiving it again.

**Note:** If a SYSMOD appears more than once in the SMPPTFIN data set, the first occurrence may be received. However, none of the subsequent versions of the SYSMOD are received, even if their rework level is higher than the one for the first version of the SYSMOD. (Message GIM40001 is issued for each of the subsequent versions of the SYSMOD.)

**RFDSNPFX**

identifies to SMP/E the prefix used in the relative file data set names for this SYSMOD. SMP/E uses this prefix when allocating data set names for the SYSMOD's relative files during RECEIVE processing.

- This operand can be specified only if the FILES operand is also specified.
- The RFDSNPFX value specified on the MCS statement must match the actual prefix used in the data set names for the associated relative files.

For example, if the names of the relative files created for a SYSMOD start with "IBM," as in `IBM.sysmod_id.F1`, the header MCS statement for the SYSMOD must specify `RFDSNPFX(IBM)` so SMP/E knows which prefix to use when allocating the data set names for the SYSMOD's relative files during RECEIVE processing.

- Following standard data set naming conventions, the prefix can be from 1 to 8 alphanumeric or national (\$, #, @) characters or a dash (-).

To enable full RACF\* protection for tape data sets and to keep the tape header within the 17-character limit (including periods), you should limit the prefix to 1 to 3 characters. If the name exceeds the 17-character limit, only the rightmost 17 characters are written to the tape header label.

*sysmod\_id*

specifies a unique 7-character system modification identifier for the APAR fix. See "Naming Conventions for SYSMODs" on page 809 for more information.

**Example: ++APAR with ++ZAP**

Here is an example of a SYSMOD containing a ++APAR statement for a temporary fix to module IFBMOD01. As the example shows, this fix is needed to answer APAR AZ12345 on an MVS system. The module must be at the service level provided by PTF UZ00004 for function FXY1040.

```
++APAR(AZ12345)          /* APAR type fix          */.
++VER(Z038)  FMID(FXY1040) /* for MVS product FXY1040 */
                   PRE(UZ00004) /* at this service level. */.
++ZAP(IFBMOD01)         /* Change to one module   */.
                   DISTLIB(AOSFB) /* in this DLIB.         */.
...
... IMASPZAP control statements
...
```

## ++ASSIGN MCS

The ++ASSIGN MCS assigns a source identifier (source ID) to one or more specified SYSMODs, as long as those SYSMODs are in the SMPPTS data set by the end of RECEIVE processing.

### Syntax

```

++ASSIGN MCS
  ►► ++ASSIGN—SOURCEID(source_id)—TO(—sysmod_id—)—◄◄
  
```

### Operands

#### SOURCEID

is a 1- to 8-character string identifying the source of the SYSMODs being processed. SMP/E associates the SOURCEID value with the SYSMODs named on the ++ASSIGN MCS.

#### TO

specifies the SYSMODs with which the source ID is to be associated.

### Usage Notes

- The source ID specified on the ++ASSIGN statement is added to any source ID that was assigned to a specified SYSMOD by the RECEIVE command. It is also added to any source IDs currently associated with a specified SYSMOD that has already been received.
- ++ASSIGN statements are processed only when the SMPPTFIN data set is processed.
  - If the whole SMPPTFIN data set is processed, all ++ASSIGN statements are processed.
  - If only selected SYSMODs are processed, the ++ASSIGN statements for those SYSMODs are processed.

++ASSIGN statements are not processed when only the SMPHOLD data set is being processed.

- The SOURCEID and TO values must be validly specified and cannot be blank or null. For more information about source ID naming conventions, see “Naming Conventions for Source IDs” on page 808.
- The source ID is not assigned to any SYSMODs that are not in the SMPPTS.
- The same SYSMOD cannot appear more than once on a single ++ASSIGN MCS.
- If the same SYSMOD appears on more than one ++ASSIGN MCS, all the source IDs are associated with the SYSMOD.
- The ++ASSIGN MCS is used in the SMPPTFIN data set and can be placed between, before, or after SYSMODs.

It must be followed by one of the following: a ++APAR, ++FUNCTION, ++PTF, or ++USERMOD MCS; another ++ASSIGN MCS; or an end-of-file. If one of

these does not follow, SMP/E does not receive the SYSMOD being processed, and it skips the ++ASSIGN MCS.

## Example: Assigning Source IDs to Service in an ESO

Here are some examples of ++ASSIGN statements for SYSMODs from several preventive service levels that have been merged into the same ESO tape. A ++ASSIGN MCS has been placed between the groups of SYSMODs to identify their source:

```
++ASSIGN SOURCEID(PUT9301) /* service level 9301. */
TO(UZ12345,UZ12346).
++PTF(UZ12345) /* PTF UZ12345 */.
++VER(Z038) FMID(HXP1100) /* for MVS function HXP1100.*/.
++MOD(A) DISTLIB(DN554) /* Update module A. */.
A
++PTF(UZ12346) /* PTF UZ12346 */.
++VER(Z038) FMID(HXP1200) /* for MVS function HXP1200.*/.
++MOD(C) DISTLIB(DN554) /* Update module C. */.
C
.
.
.

++ASSIGN SOURCEID(PUT9302) /* service level 9302. */
TO(UZ12347,UZ12348).
++PTF(UZ12347) /* PTF UZ12347 */.
++VER(Z038) FMID(HXP1100) /* for MVS function HXP1100.*/.
++IF FMID(HXP1200) THEN REQ(UZ12348).
++MOD(D) DISTLIB(DN554) /* Update module D. */.
D
++PTF(UZ12348) /* PTF UZ12348 */.
++VER(Z038) FMID(HXP1200) /* for MVS function HXP1200.*/.
++MOD(A) DISTLIB(AOS12) /* Update module A. */.
A
++MOD(B) DISTLIB(AOS12) /* Update module B. */.
B
.
.
.
```

## Data Element MCS

Data element MCSs describe elements that are not macros, modules, or source. Data elements can have any of the following characteristics:

- A data element must be a member of a partitioned data set (DSORG=PO).
- The record format (RECFM) must be F, FA, FM, FB, FBA, FBM, V, VA, VM, VB, VBA, or VBM.
- Elements with fixed-length records can have any logical record length (LRECL). They are not restricted to an LRECL of 80.
- Elements with variable-length records cannot contain spanned records.
- The maximum LRECL for a data element is 32,654.
- The records can be numbered or unnumbered.
- A VSAM data set can be a data element if it is in REPRO format. However, after the data is installed by SMP/E, the user also has to run an AMS REPRO job to create the original form of the VSAM data.

There are MCSs to replace data elements, just as there are MCSs to replace other types of elements. (There are no MCSs to update data elements.) Table 26 shows the MCSs that can be used for data elements.

*Table 26 (Page 1 of 2). MCS Statements for Data Elements. If an element is provided in only one language, the x's can be left off the MCS. If an element is provided in more than one language, replace the x's with the appropriate value from Table 27 on page 526.*

MCS	Description
++BOOKxxx	Online book member
++BSINDxxx	Index for an online publications library (bookshelf)
++CGMxxx	Graphics source for an online book
++CLIST	CLIST
++DATA	Data not covered by other types
++DATA1--++DATA5	IBM generic data types 1–5 These are for IBM use only, to define elements that are not covered by any existing data types.
++DATA6xxx	IBM generic data type 6 This is for IBM use only to define an element not covered by any existing data types.
++EXEC	EXEC
++FONTxxx	Printer Font Object Contents Architecture (FOCA) font
++GDFxxx	GDF graphics panel
++HELPxxx	Help information (for example, a member in SYS1.HELP or a dialog help panel)
++IMGxxx	Graphics image for an online book
++MSGxxx	Message member (such as for a dialog or for a message data set)

*Table 26 (Page 2 of 2). MCS Statements for Data Elements. If an element is provided in only one language, the x's can be left off the MCS. If an element is provided in more than one language, replace the x's with the appropriate value from Table 27 on page 526.*

MCS	Description
++PARM	PARMLIB member
++PNLxxx	Panel for a dialog
++PROBJxxx	Printer object element
++PROC	Procedure in PROCLIB
++PRSRCxxx	Printer source element
++PSEGxxx	Graphics page segment for an online book
++PUBLBxxx	Online publications library (bookshelf)
++SAMPxxx	Sample data, program, or JCL in a data set for sample code
++SKLxxx	File skeleton for a dialog
++TBLxxx	Table for a dialog
++TEXTxxx	PDS member containing text plus SCRIPT tags
++USER1–++USER5	User-defined data types 1–5 These are for user-defined elements that are not covered by any existing data types.
++UTINxxx	General utility input
++UTOUTxxx	General utility output

## Supporting Several Languages

Some types of elements, such as panels, messages, or text, may have to be translated into several languages. In these cases, the corresponding MCSs contain xxx to indicate which language is supported by a given element. Figure 84 shows an example where product XX1 must provide both English and French support for a message module, a panel, a panel message, and a sample element.

++FUNCTION(FXX1101).		++FUNCTION(FXX1102).
++VER(Z038) FMID(EXX1100).		++VER(Z038) FMID(EXX1100).
++MOD(ZZZMOD0E)... message		++MOD(ZZZMOD0F)...
DISTLIB(AZZZMOD1). modules		DISTLIB(AZZZMOD1).
++PNLENU(ZZZPNL01)... panels		++PNLFRA(ZZZPNL01)...
DISTLIB(AZZZPNLE)		DISTLIB(AZZZPNLF)
SYSLIB(SZZZPNLE).		SYSLIB(SZZZPNLF).

Figure 84 (Part 1 of 2). Example of Using Data Element MCSs



```

++MSGENU(ZZZMSG01)...  dialog  ++MSGFRA(ZZZMSG01)...
DISTLIB(AZZZMSGE)     messages DISTLIB(AZZZMSGF)
SYSLIB(SZZZMSGE).     SYSLIB(SZZZMSGF).

++SAMPENU(ZZZSMP01)... samples ++SAMPFRA(ZZZSMP01)...
DISTLIB(AZZZSAME)     DISTLIB(AZZZSAMF)
SYSLIB(SZZZSAME).     SYSLIB(SZZZSAMF).

```

Figure 84 (Part 2 of 2). Example of Using Data Element MCSs

**Notes:**

1. The message modules can be in the same distribution library, because the element names are different.
2. For the panels, dialog messages, and samples, there is a different element *type* for each language version of an element. Therefore, the element *name* can remain the same for all the languages in which the element is supported. However, elements with the same name must be installed in different libraries. (SMP/E does not check whether different types of data elements have the same name. Likewise, SMP/E does not prevent elements with the same name from being installed in the same libraries.)

Table 27 on page 526 shows the xxx values that can be used when the MCS indicates the language being supported.

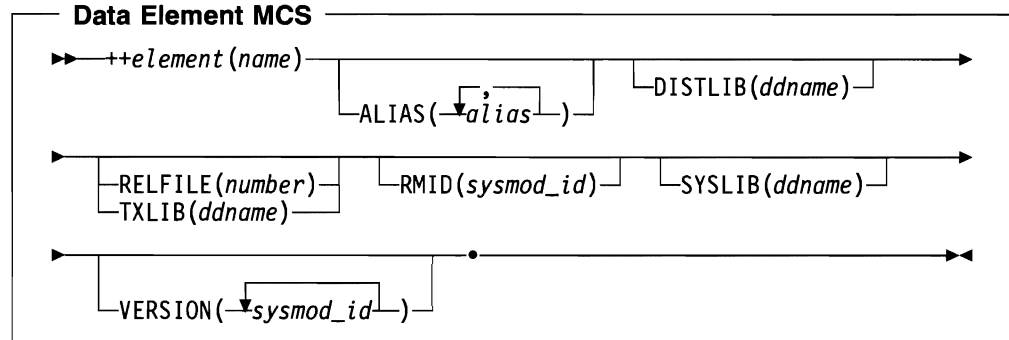
Value	Language	Value	Language
ARA	Arabic	HEB	Hebrew
CHS	Simplified Chinese	ISL	Icelandic
CHT	Traditional Chinese	ITA	Italian (Italy)
DAN	Danish	ITS	Italian (Switzerland)
DES	German (Switzerland)	JPN	Japanese
DEU	German (Germany)	KOR	Korean
ELL	Greek	NLB	Dutch (Belgium)
ENG	English (United Kingdom)	NLD	Dutch (Netherlands)
ENP	Uppercase English	NOR	Norwegian
ENU	English (United States)	PTB	Portuguese (Brazil)
ESP	Spanish	PTG	Portuguese (Portugal)
FIN	Finnish	RMS	Rhaeto-Romanic
FRA	French (France)	RUS	Russian
FRB	French (Belgium)	SVE	Swedish
FRC	French (Canada)	THA	Thai
FRS	French (Switzerland)	TRK	Turkish

## Syntax

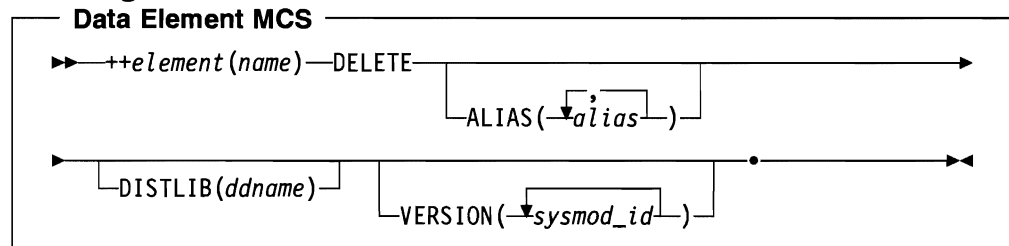
The syntax to be used depends on the processing to be done for the element:

- Adding or replacing the element
- Deleting the element

### Adding or Replacing a Data Element



### Deleting a Data Element



## Operands

### ALIAS

specifies the alias names for the data element in both the target and distribution libraries.

You can use ALIAS when data elements of the same type must be defined in the same zone and must have the same name for programming access. In this case, you can specify the common name on ALIAS and a unique name as the data element name.

### DELETE

indicates that the element is to be removed from the target libraries, the distribution libraries, and the SMP/E data sets.

**Note:** DELETE is mutually exclusive with RELFILE, RMID, SYSLIB, and TXLIB.

**DISTLIB**

specifies the ddname of the distribution library for the data element.

**Notes:**

1. **DISTLIB** must be specified if the element has not been previously recorded on the target zone or distribution zone.
2. If a data element entry already exists in the target zone or distribution zone and the value currently in that entry does not match that specified in the **DISTLIB** operand, the **SYSMOD** is not applied or accepted.

*element*

specifies the type of element. Table 26 on page 523 shows the MCSs used for the various element types.

*name*

specifies the name of the element to be replaced. The name can contain any alphanumeric characters and \$, #, @, or hex C0.

**RELFILE**

specifies which relative file associated with the **SYSMOD** contains this element. This operand is required if you provide the element in **RELFILE** format, rather than inline or in a **TXLIB** data set.

**Note:** **RELFILE** is mutually exclusive with **DELETE** and **TXLIB**.

**RMID**

specifies the last PTF that **replaced** this data element. This operand can be used only in a service-updated function, and the specified PTF must be integrated into the function.

**Note:** **RMID** is mutually exclusive with **DELETE**.

**SYSLIB**

specifies the ddname of the target library for the specified element.

**Note:** **SYSLIB** is mutually exclusive with **DELETE**.

**TXLIB**

specifies the ddname of the partitioned data set containing the element. This operand is required if the element is provided in a **TXLIB** data set rather than inline or in **RELFILE** format.

**Notes:**

1. **SMPTLIB** cannot be used as a value on the **TXLIB** operand.
2. **TXLIB** is mutually exclusive with **DELETE** and **RELFILE**.

**VERSION**

specifies one or more function **SYSMODs** that currently contain the element. The function containing the data element statement takes over ownership of the element from the specified functions.

When **VERSION** is specified on an element statement, it overrides any **VERSION** operand values specified on the **++VER** MCS.

## Usage Notes

- If the element is packaged inline, it must immediately follow the data element MCS and must not contain any records that start with the characters ++. Neither **RELFILE** nor **TXLIB** can be specified on the data element MCS.
- To be packaged inline, a data element must contain fixed-block-80 records. If the original format of the element is not fixed-block-80 records, you can use GIMDTS to transform the element into the required format before packaging it. Later, when SMP/E installs the element, it is changed back to its original format. For more information about using GIMDTS, see Chapter 37.
- If the data element is packaged in a TXLIB data set, the ddname specified on the TXLIB operand is required during APPLY and ACCEPT processing.
- For information about packaging SYSMODs in RELFILE, TXLIB, or inline format, see the *Standard Packaging Rules for MVS-Based Products* manual.

## Example: Packaging a CLIST in a Function

Here is an example of a SYSMOD containing a ++CLIST statement to install your CLIST and have SMP/E track any changes to it. It can be packaged as a function, as shown below:

```
++FUNCTION(MYCLST1)      /* Function.          */
++VER(Z038)             /* For MVS systems.  */
++CLIST(CLIST1)         /* Install this CLIST
                        TXLIB(NEWSMP) /* from this TXLIB
                        DISTLIB(AMACLIB) /* into this DLIB and
                        SYSLIB(ISPCLIB) /* this target library. */
```

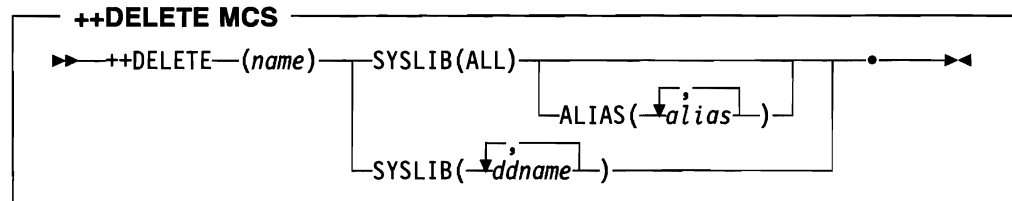
## ++DELETE MCS

During APPLY processing, the ++DELETE MCS deletes a load module and any known alias names from a target library and the associated target zone. It can also delete alias names without deleting the load module itself.

During ACCEPT processing, if inline JCLIN was processed, the distribution zone's entries are updated to reflect the ++DELETE changes. The target libraries and target zone entries are not updated.

**Note: This change is not reversible.** You cannot restore the SYSMOD containing the ++DELETE MCS; therefore, you cannot recreate the deleted load module.

## Syntax



## Operands

### ALIAS

specifies that only the indicated alias names are to be deleted, but not the load module.

- An alias name can have up to 64 characters.

Although a load module residing in a PDSE can have an alias name greater than 8 characters, the ++DELETE statement cannot be used to delete such an alias value. To delete such an alias value without deleting the load module, you need to resupply JCLIN to define the load module without providing an ALIAS statement for the alias value to be deleted. Make sure to also include a ++MOD statement for a module in the load module to force SMP/E to relink the load module.

- Single apostrophes must be used as delimiters for an alias name in these cases:
  - The alias name contains lowercase alphabetic characters.
  - The alias name contains a character that is not uppercase alphabetic, numeric, national (\$, #, or @), slash (/), plus (+), hyphen, period, or ampersand (&).
  - The alias name is continued on another line in the control statement.

Apostrophes used as delimiters do not count as part of the 64-character limit.

- If an apostrophe is part of the alias name and not a delimiter, it needs to be doubled. These two apostrophes count as 2 characters in the 64-character limit.

- Characters specified must be in the range X'41' through X'FE'. Unlike the binder, SMP/E does not support shift-in (X'0F') and shift-out (X'0E') characters.
- When processing the ++DELETE statement, SMP/E uses the alias name as is, and does not enforce any rules the binder might be using as a result of the CASE execution parameter.

### Notes:

1. **Do not** specify this operand if you also want to delete the load module. ++DELETE without the ALIAS operand automatically deletes any alias names associated with the load module.
2. When **ALIAS** is specified, SMP/E checks the ALIAS control statements in the LMOD entry to verify that the specified alias name is actually an alias of the load module. For copied load modules, instead of looking for ALIAS control statements, SMP/E checks the corresponding MOD entry's TALIAS subentries.
3. If a valid **ALIAS** value is specified, SMP/E deletes the alias from all known target libraries, no matter what was coded for SYSLIB. SMP/E overrides the SYSLIB value with SYSLIB(ALL).
4. When you specify **ALIAS** to delete an alias for a load module, you must reflect this change using JCLIN. To do this, include a ++JCLIN statement with JCLIN data containing a link-edit step for the load module, with the alias deleted from the list of aliases on the link-edit ALIAS statement. This causes SMP/E to replace the alias list in the LMOD entry.

### *name*

specifies the name of the load module to be deleted.

### **SYSLIB**

specifies the ddname of the target library where the load module resides.

- If **ALL** is specified, the load module is deleted from all target libraries defined in the target zone.
- If a single ddname is specified, the load module is deleted only from the indicated target library.

If the load module or alias is to be deleted from two target libraries, a second ddname can be specified. However, to make sure the module is deleted from both libraries, you should use **ALL** instead.

- For load modules with a SYSLIB allocation (or load modules having a CALLLIBS subentry), specifying **ALL** or deleting the load module from all the target libraries in which it resides, deletes the base version of the load module from the SMPLTS library.

**Note:** If **ALIAS** is specified, SMP/E deletes the alias from all known target libraries, no matter what was coded for SYSLIB. SMP/E overrides the SYSLIB value with SYSLIB(ALL).

## Usage Notes

- **SYSLIB** must always be specified to identify the affected target libraries.
- **ALIAS** should be specified **only** if alias names, and not load modules, are to be deleted.
- ++DELETE statements must follow any ++VER and ++IF statements and must precede any ++JCLIN or element MCSs.
- Regardless of the order in which ++MOVE, ++RENAME, and ++DELETE statements are coded in a SYSMOD, they are always processed in this order for APPLY and ACCEPT:
  1. ++MOVE
  2. ++RENAME
  3. ++DELETE

Afterwards, the ++JCLIN statements are processed, and then the element statements are processed.

**Note:** You cannot restore the SYSMOD containing the ++DELETE MCS.

## Examples

The following examples are provided to help you use the ++DELETE MCS:

- “Example 1: Deleting a Single Load Module”
- “Example 2: Deleting an Alias from a Load Module” on page 532
- “Example 3: Deleting an Alias from a Load Module in a Hierarchical File System” on page 532

### Example 1: Deleting a Single Load Module

Here is an example of a SYSMOD containing a ++DELETE statement that deletes load module LMODA from SYS1.LINKLIB:

```

++PTF(UR01234)          /* Identify the PTF number. */.
++VER(Z038) FMID(HXY1300) /* For MVS function HXY1300.*/.
++IF (ESY1300) THEN     /* If ESY1300 is installed */
  REQ(UR12399)          /* UR12399 is required. */.
++HOLD(UR01234)         /* Hold UR01234. */
                        FMID(HXY1300) /* For MVS function HXY1300.*/
                        SYSTEM          /* System hold */
                        REASON(DELETE) /* because of ++DELETE. */
                        COMMENT(THIS DELETION OF LMODA FROM LINKLIB
                                IS IRREVERSIBLE).
++DELETE (LMODA)        /* Delete load module LMODA */
  SYSLIB(LINKLIB)       /* from LINKLIB. */.
++JCLIN                 /* JCLIN follows. */.
.
.
.
++MOD(MODAA) DISTLIB(AOSXX) /* Element MCS statements. */.

```

**Example 2: Deleting an Alias from a Load Module**

Assume that IBM has shipped you a PTF that deletes an alias for load module LMODA. Here is an example of a SYSMOD containing a ++DELETE statement that deletes an alias from LMODA:

```

++PTF(UR01235)          /* Identify the PTF number. */.
++VER(Z038) FMID(HXY1300) /* For MVS function HXY1300.*/.
++HOLD(UR01235)         /* Hold UR01235. */
                        FMID(HXY1300) /* For MVS function HXY1300.*/
                        SYSTEM         /* System hold */
                        REASON(DELETE) /* because of ++DELETE. */
                        COMMENT(THE DELETION OF THE ALIAS FOR LMODA
                                IS IRREVERSIBLE).
++DELETE (LMODA)        /* Identify LMOD LMODA. */
  SYSLIB(ALL)           /* Process all SYSLIBs. */
  ALIAS(OTHNAME)       /* Identify alias. */.
++JCLIN                /* JCLIN to delete alias. */.
  .
  .
  .

```

**Example 3: Deleting an Alias from a Load Module in a Hierarchical File System**

Assume that IBM has shipped you a PTF that deletes an alias for load module BPXLMOBDB, which resides in a hierarchical file system. Here is an example of a SYSMOD containing a ++DELETE statement that deletes an alias from BPXLMOBDB:

```

++PTF(UZ00440)          /* Identify the PTF number. */.
++VER(Z038) FMID(HOP1101) /* For MVS function HOP1101.*/.
++HOLD(UZ00440)         /* Hold UZ00440. */
                        FMID(HOP1101) /* For MVS function HOP1101.*/
                        SYSTEM         /* System hold */
                        REASON(DELETE) /* because of ++DELETE. */
                        COMMENT(THE DELETION OF THE ALIAS FROM
                                BPXLMOBDB IS IRREVERSIBLE).
++DELETE (BPXLMOBDB)   /* Identify LMOD BPXLMOBDB. */
  SYSLIB(ALL)           /* Process all SYSLIBs. */
  ALIAS('./nickname')  /* Identify alias. */.
++JCLIN                /* JCLIN follows. */.
  .
  .
  .
++MOD(BPXMODAA) DISTLIB(AOSXX) /* Element MCS statements. */.

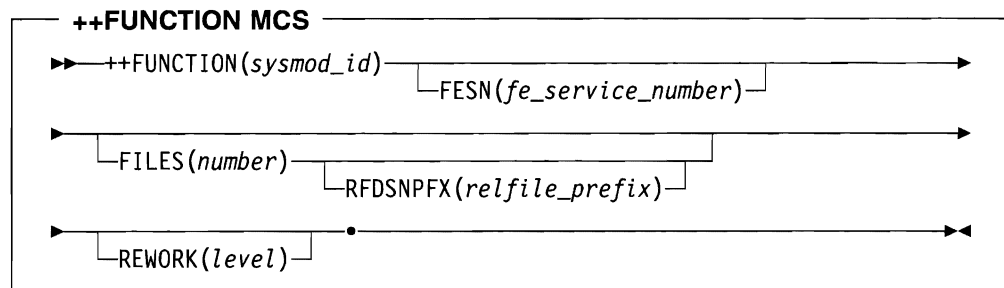
```



## ++FUNCTION MCS

The ++FUNCTION MCS identifies a SYSMOD as a base function or dependent function. This type of SYSMOD introduces a new or replacement function into target system and distribution libraries. All other MCSs follow this header MCS statement. For more information about packaging a function, see the *Standard Packaging Rules for MVS-Based Products* manual.

### Syntax



### Operands

#### FESN

is a 7-character field engineering (FE) service number.

#### FILES

specifies the number of relative files belonging to this function. It can be a decimal number from 1 to 9999, and is used only if the function is packaged in relative files, rather than inline or in indirect libraries. For information about packaging SYSMODs in relative files, see the *Standard Packaging Rules for MVS-Based Products* manual.

#### Notes:

1. Functions are generally packaged in relative files to improve SMP/E's performance when applying and accepting the SYSMOD.
2. If a packager uses a high-level qualifier on RELFILE data sets, the RFDSNPFX operand on the header MCS (not the RFPREFIX operand on the RECEIVE command) **must** be used to identify that high-level qualifier.

#### REWORK

is the level of this SYSMOD, which was reworked for minor changes. Up to eight numeric characters can be specified.

REWORK is generally used only for SYSMODs supplied by IBM that have been reworked for minor changes, such as for a service update or to use a ++MOVE, ++RENAME, or ++DELETE MCS. For these SYSMODs, the REWORK level is *yyyddd*, which is the year followed by the Julian date (for example, 1990110).

REWORK allows an updated SYSMOD to be automatically received again, as long as it is more recent than the version that has already been received. This takes the place of rejecting the SYSMOD and receiving it again.

**Note:** If a SYSMOD appears more than once in the SMPPTFIN data set, the first occurrence may be received. However, none of the subsequent versions of the SYSMOD are received, even if their rework level is higher than the one for the first version of the SYSMOD. (Message

GIM40001 is issued for each of the subsequent versions of the SYSMOD.)

**RFDSNPFXX**

identifies to SMP/E the prefix used in the relative file data set names for this SYSMOD. SMP/E uses this prefix when allocating data set names for the SYSMOD's relative files during RECEIVE processing.

- This operand can be specified only if the FILES operand is also specified.
- The RFDSNPFXX value specified on the MCS statement must match the actual prefix used in the data set names for the associated relative files.

For example, if the names of the relative files created for a SYSMOD start with "IBM," as in IBM.*sysmod\_id*.F1, the header MCS statement for the SYSMOD must specify RFDSNPFXX(IBM) so SMP/E knows which prefix to use when allocating the data set names for the SYSMOD's relative files during RECEIVE processing.

- Following standard data set naming conventions, the prefix can be from 1 to 8 alphanumeric or national (\$, #, @) characters or a dash (-).

To enable full RACF protection for tape data sets and to keep the tape header within the 17-character limit (including periods), you should limit the prefix to 1 to 3 characters. If the name exceeds the 17-character limit, only the rightmost 17 characters are written to the tape header label.

*sysmod\_id*

is a unique 7-character name, or SYSMOD ID, for the function. This ID is also called a function modification identifier (FMID). For more information, see "Naming Conventions for SYSMODs" on page 809.

**Usage Notes**

A function cannot contain statements that update elements (++MACUPD, ++SRCUPD, and ++ZAP).

**Example: ++FUNCTION in RELFILE Format**

Here is an example of a function SYSMOD to be created with a SYSMOD ID of JXY1040 that is dependent on function JXY1000. The elements and JCL input data are members of three unloaded partitioned data sets on a tape created using the relative file method:

```

++FUNCTION(JXY1040)          /* New function SYSMOD */
      FILES(3)              /* in RELFILE format. */
      RFDSNPFXX(IBM)        /* RELFILE prefix for IBM. */.
++VER(Z038) FMID(JXY1000)  /* For MVS function JXY1000.*/.
++JCLIN RELFILE(1)         /* JCLIN in RELFILE 1. */.
++MOD(IFBMOD01)            /* This module */
      DISTLIB(AOSFB)       /* for this DLIB */
      RELFILE(2)           /* is in RELFILE 2. */.
++MOD(IFBMOD02)            /* This module */
      DISTLIB(AOSFB)       /* for this DLIB */
      RELFILE(2)           /* is in RELFILE 2. */.
++MAC(IFBMAC01)            /* This macro */
      DISTLIB(AOSMAC)      /* for this DLIB */
      RELFILE(3)           /* is in RELFILE 3. */.

```

## ++HFS MCS

The ++HFS MCS describes a single replacement for an element residing in a hierarchical file system (HFS). (There are no MCSs to update HFS elements.) HFS elements can have any of the following characteristics:

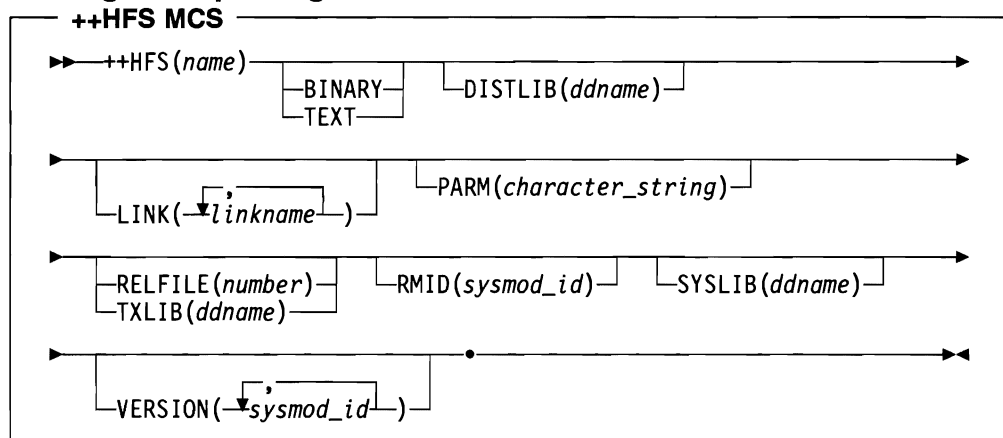
- The record format (RECFM) must be F, FA, FM, FB, FBA, FBM, V, VA, VM, VB, VBA, or VBM.
- Elements with variable-length records cannot contain spanned records.
- The maximum LRECL is 32,654.
- The records can be numbered or unnumbered.

## Syntax

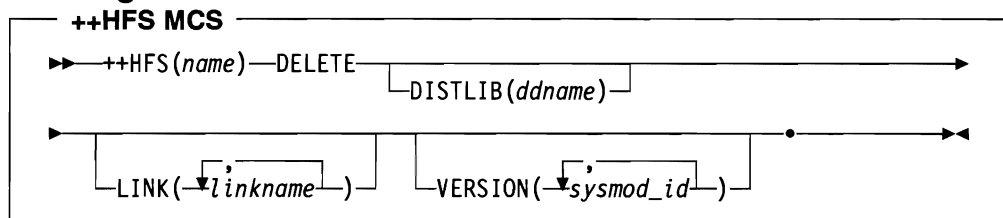
The syntax to be used depends on the processing to be done for the element:

- Adding or replacing the element
- Deleting the element

### Adding or Replacing an HFS Element



### Deleting an HFS Element



## Operands

### BINARY

indicates that the hierarchical file system copy utility should install the element into the HFS in *binary* mode. This means that the element is installed in its entirety as a data stream, with no breaks for logical records.

**Notes:**

1. BINARY is mutually exclusive with TEXT.
2. When BINARY is specified on the ++HFS statement, SMP/E sets the BINARY mode indicator in the HFS entry. When TEXT is specified on the ++HFS statement, SMP/E sets the TEXT mode indicator in the HFS entry.

If neither BINARY nor TEXT is specified on the ++HFS statement, SMP/E uses the mode indicator in the HFS entry to tell the HFS copy utility how to install the element.

If neither BINARY nor TEXT is specified on the ++HFS statement and there is no mode indicator in the HFS entry, the HFS copy utility determines how to install the element.

**DELETE**

specifies that the HFS element is to be removed from the “target library” (hierarchical file system) and the distribution library.

**Notes:**

1. DELETE is mutually exclusive with all other operands except DISTLIB, LINK, and VERSION.
2. If the element statement is in a base function, you may want to use the DELETE operand on the ++VER MCS to delete the previous release, rather than on the element statement to delete a specific element.

**DISTLIB**

specifies the ddname of the distribution library for the specified HFS element. During ACCEPT processing, SMP/E installs the HFS element into the distribution library as a member. (The distribution library must be a PDS or PDSE; it cannot be part of the hierarchical file system.)

**Notes:**

1. **DISTLIB** must be specified when the HFS element is first installed.
2. If a HFS entry already exists in the target zone or distribution zone and the value currently in that entry does not match that specified in the DISTLIB operand, the SYSMOD is not applied or accepted.

**LINK**

specifies the alternative names by which this HFS element can be known in the hierarchical file system. Each linkname is passed to the HFS copy utility as an execution parameter.

**Notes:**

1. The linkname can be from 1 to 64 characters.
2. A linkname can be enclosed in single apostrophes ('). A linkname **must** be enclosed in single apostrophes if any of the following is true:
  - The linkname contains lowercase alphabetic characters.
  - The linkname contains a character that is not uppercase alphabetic, numeric, national (\$, #, or @), slash (/), plus (+), hyphen, period, or ampersand (&).
  - The linkname spans more than one line in the control statement.

The single apostrophes used to enclose a linkname (the delimiters) do not count as part of the 64-character limit.

3. Any apostrophes specified as part of a linkname (not the delimiters) must be doubled.

Double apostrophes count as two characters in the 64-character limit.

4. The linkname can include characters X'40' through X'FE'.

5. LINK values are saved and passed to the HFS copy utility as follows:

- If LINK is specified on the ++HFS statement, any values previously saved in the HFS entry are overlaid.
- If LINK is not specified on the ++HFS statement and saved values exist in the HFS entry, the saved values are passed to the HFS copy utility as execution parameters.

If LINK is not specified on the ++HFS statement and there are no saved values in the HFS entry, no linknames are passed to the HFS copy utility.

*name*

specifies the name of the HFS element member. The name can contain any uppercase alphabetic, numeric, or national (\$, #, @) character and can be 1 to 8 characters long.

**PARM**

specifies a character string that is to be passed to the hierarchical file system copy utility as an execution-time parameter. The maximum length of this character string is 300 bytes of nonblank data. If any blanks are specified in the PARM value, they are deleted by SMP/E during processing and do not count toward the 300-byte maximum.

**Notes:**

1. PARM is an optional operand.
2. The character string can be entered free-form, without regard to blanks (which are compressed out of the string), and can span multiple 80-byte records.
3. If parentheses are specified in the PARM value, there must always be a pair (left and right); otherwise, the results are unpredictable.
4. PARM values are saved and passed to the HFS copy utility as follows:
  - If PARM is specified on the ++HFS statement, any values previously saved in the HFS entry are overlaid.
  - If PARM is not specified on the ++HFS statement and saved values exist in the HFS entry, the saved values are passed to the HFS copy utility as execution parameters.

If PARM is not specified on the ++HFS statement and there are no saved values in the HFS entry, no parameters are saved from the ++HFS statement or passed from the HFS entry to the HFS copy utility.

5. If the UTILITY entry for the HFS copy utility specifies a PARM value, those parameters are passed to the utility in addition to any parameters saved in the HFS entry.

### RELFILE

identifies which relative file associated with the SYSMOD contains this element. This operand is required if you provide the element in RELFILE format, rather than inline or in a TXLIB data set.

#### Notes:

1. The RELFILE value must be a decimal number from 1 to 9999.
2. RELFILE is mutually exclusive with TXLIB.

### RMID

specifies the last SYSMOD that **replaced** this HFS element. This operand can be used only in a service-updated function, and the specified PTF must be integrated into the function.

### SYSLIB

specifies the ddname of the “target library” within the hierarchical file system for the HFS element.

During APPLY processing, the HFS copy utility installs the HFS element into the hierarchical file system. During RESTORE processing, the HFS copy utility copies the HFS element from the distribution library member into the hierarchical file system.

**Note:** SYSLIB must be specified when the HFS element is first installed.

### TEXT

indicates that the hierarchical file system copy utility should install the element into the hierarchical file system (HFS) in *text* mode. This means that the element is installed with breakpoints for logical records.

#### Notes:

1. TEXT is mutually exclusive with BINARY.
2. When TEXT is specified on the ++HFS statement, SMP/E sets the TEXT mode indicator in the HFS entry. When BINARY is specified on the ++HFS statement, SMP/E sets the BINARY mode indicator in the HFS entry.

If neither BINARY nor TEXT is specified on the ++HFS statement, SMP/E uses the mode indicator in the HFS entry to tell the HFS copy utility how to install the element.

If neither BINARY nor TEXT is specified on the ++HFS statement and there is no mode indicator in the HFS entry, the HFS copy utility determines how to install the element.

### TXLIB

is the ddname of the partitioned data set containing the HFS element. This operand is required if the HFS element is provided in a data set that the users have access to, rather than inline or in RELFILE format.

#### Notes:

1. SMPTLIB cannot be used as a value on the TXLIB operand.
2. TXLIB is mutually exclusive with LKLIB and RELFILE.

**VERSION**

specifies one or more function SYSMODs that currently contain the element. The function containing the ++HFS MCS takes over ownership of the element from the specified functions.

When **VERSION** is specified on an element statement, it overrides any **VERSION** operand values that might be specified on the ++VER MCS.

**Usage Notes**

- If the HFS element is packaged inline, it must immediately follow the ++HFS MCS and must not contain any records starting with ++. Neither **RELFILE** nor **TXLIB** can be specified on the ++HFS MCS.
- To be packaged inline, an HFS element must contain fixed-block-80 records. If the original format of the element is not fixed-block-80 records, you can use **GIMDTS** to transform the element into the required format before packaging it. Later, when SMP/E installs the element, it is changed back to its original format. For more information about using **GIMDTS**, see Chapter 37.
- If the HFS element is packaged in a **TXLIB** data set, the ddname specified in the **TXLIB** operand is required during **APPLY** and **ACCEPT** processing.
- For information about elements packaged in **RELFILE** format, see the *Standard Packaging Rules for MVS-Based Products* manual.

**Example: Packaging an HFS Element in a Function**

Here is an example of a SYSMOD containing a ++HFS statement to install your element and have SMP/E track any changes to it. The element is to be installed in **TEXT** mode and can also be known by its linkname "USERHFS." It can be packaged as a function, as shown below:

```

++FUNCTION(MYHFSEL)          /* Function.                */
++VER(Z038)                  /* For MVS systems.        */
++HFS(HFSELEM1)             /* Install this HFS element */
                             /* from this TXLIB         */
                             /* into this DLIB and      */
                             /* this target library.    */
                             /* Use TEXT mode.          */
                             /* Define linkname.        */
TXLIB(NEWHFS)                /*                          */
DISTLIB(ABPXLIB)             /*                          */
SYSLIB(BPXLIB1)             /*                          */
TEXT                          /*                          */
LINK(USERHFS)                /*                          */

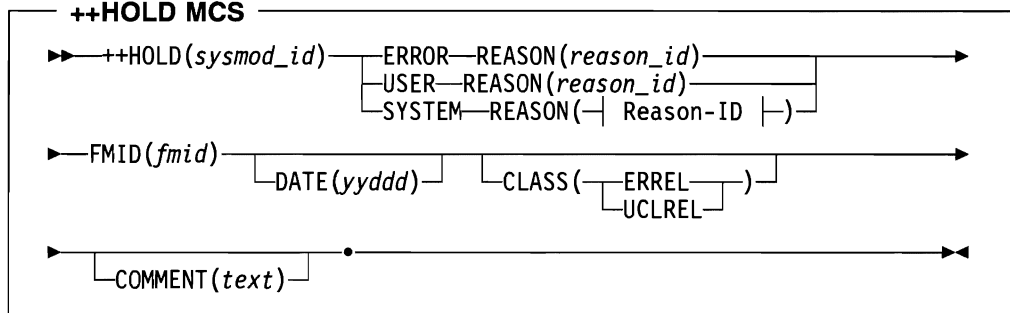
```

To install the function, you need to specify **DDDEF** entries or **DD** statements for the **TXLIB**, **DISTLIB**, and **SYSLIB** data sets. Remember that the **SYSLIB** data set is actually a pathname in the hierarchical file system. For an example of defining a pathname in a **DDDEF** entry, see "Example 10: Defining Pathnames in a Hierarchical File System" on page 669.

## ++HOLD MCS

The ++HOLD MCS identifies a SYSMOD to be placed into exception SYSMOD status (signifying that special SMP/E processing is required before it can be applied or accepted). ++HOLD statements can occur within a SYSMOD (internal HOLDDATA), or they can be read directly from the SMPHOLD file during RECEIVE processing (external HOLDDATA). For additional information on processing the ++HOLD statements, see "Processing ++HOLD and ++RELEASE Statements" on page 253.

### Syntax



**SYSTEM Reason IDs Used by IBM:**

ACTION
AO
DELETE
DEP
DOC
EC
EXRF
FULLGEN
IOGEN
MSGSKEL
MVSCP

### Operands

#### CLASS

a 1- to 7-character string indicating an alternative reason to release an exception SYSMOD for processing. A class name is specified along with a reason ID to identify a condition when the reason ID need not be resolved. For example, a SYSMOD may require a UCLIN change unless a SYSGEN is done. The ++HOLD MCS in this SYSMOD has a HOLDSYSTEM reason ID of UCLIN and a HOLDCLASS name of SYSGEN. The same class name can be specified on any number of ++HOLD statements in any number of SYSMODs.



These are the specific values currently used by IBM:

**Class Explanation**

**ERREL** The SYSMOD is held for an error reason ID but should be installed anyway. IBM has determined that the problem the SYSMOD resolves is significantly more critical than the error reflected by the holding APAR.

**UCLREL** UCLIN needed for the SYSMOD has been handled by IBM and no longer requires your attention.

For additional information, see “Naming Conventions for HOLD Reason IDs and HOLD Classes” on page 807.

**COMMENT**

free-form text to be used to describe the problem identified by the REASON operand. The comments supplied are associated only with the reason ID supplied. The comments can be in single-byte characters (such as English alphanumeric characters) or double-byte characters (such as Kanji).

**DATE**

specifies the date that the ++HOLD statement was generated. The date is specified as *yyddd*, where *yy* is the last two digits of the year and *ddd* is the Julian date.

**ERROR, SYSTEM, or USER**

specifies the hold category into which the SYSMOD is to be put. At least one of the values must be specified.

**ERROR** An APAR reported an error in the SYSMOD. The SYSMOD should not be processed until the APAR is resolved. A PTF held for this reason is also called a *program error PTF*, or PE-PTF. SMP/E automatically releases the SYSMOD when a SYSMOD that supersedes the APAR is processed. Error holds can be provided in a separate HOLD file or data set, such as SMPHOLD.

**Note:** **ERROR** can also be specified as **ERR**.

**SYSTEM** Special action outside normal SMP/E processing is required for the SYSMOD. Examples are SYSMODs requiring a SYSGEN after they are installed, or SYSMODs requiring the installation of an associated engineering change (EC) level. System holds can appear in the SYSMOD itself or in a separate HOLD input file.

**Note:** **SYSTEM** can also be specified as **SYS**.

**USER** The SYSMOD requires special processing because of a decision you have made. User holds can be provided in a separate hold file or data set, such as SMPHOLD.

**FMID**

specifies the FMID to which the held SYSMOD is applicable. This information allows SMP/E to receive only those statements associated with FMIDs defined in the user's global zone. This operand is required.

**REASON**

a 1- to 7-character string used to help users identify the reason why the SYSMOD is being put into exception SYSMOD status. The reason IDs that can be specified depend on the type of hold.

**Note:** A SYSMOD can contain only one ++HOLD MCS for each required reason ID.

- An *error reason ID* is the APAR number that caused the SYSMOD to be placed in exception status.
- A *system reason ID* is a 1- to 7-character string used to identify some special processing the SYSMOD requires. These are the specific values currently used by IBM:

<b>ID</b>	<b>Explanation</b>
<b>ACTION</b>	The SYSMOD needs special handling before or during APPLY processing, ACCEPT processing, or both.
<b>AO</b>	The SYSMOD may require action to change automated operations procedures and associated data sets and user exits in products or in customer applications. The PTF cover letter describes any changes (such as to operator message text, operator command syntax, or expected actions for operator messages and commands) that can affect automation routines.
<b>DELETE</b>	The SYSMOD contains a ++DELETE MCS, which deletes a load module from the system.
<b>DEP</b>	The SYSMOD has a software dependency.
<b>DOC</b>	The SYSMOD has a documentation change that should be read before the SYSMOD is installed.
<b>EC</b>	The SYSMOD needs a related engineering change.
<b>EXRF</b>	The SYSMOD must be installed in both the active and the alternative MVS/XA* Extended Recovery Facility (XRF) systems at the same time to maintain system compatibility. (If you are not running XRF, you should bypass this reason ID.)
<b>FULLGEN</b>	The SYSMOD needs a complete system or subsystem generation to take effect.
<b>IOGEN</b>	The SYSMOD needs a system or subsystem I/O generation to take effect.
<b>MSGSKEL</b>	This SYSMOD contains message changes that must be compiled for translated versions of the message changes to become operational on extended TSO consoles.

If you want to use translated versions of the messages, you must run the message compiler once for the library containing the English message outlines, and once for each additional language you want to be available on your system. For details, see the *MVS/ESA Planning: Operations* manual.

If you want to use **only** the English version of the messages, you do not need to run the message compiler. You should bypass this reason ID.

**MVSCP** The SYSMOD requires the MVS configuration program to be run for the change to take effect.

- A *user reason ID* is defined by the user.

For additional information, see “Naming Conventions for HOLD Reason IDs and HOLD Classes” on page 807.

*sysmod\_id*

specifies that SMP/E is to put the identified SYSMOD into exception SYSMOD status. This operand is required.

## Usage Notes

- If the text within the COMMENT operand contains parentheses, they must be in matched pairs. For example:

```
COMMENT ( ADD 'PARM(COND(5))' ON THE EXEC )
```

This works, but the following statements result in an error:

```
COMMENT ( ADD OPENING PARENTHESES '(' )
```

or

```
COMMENT ( ADD CLOSING PARENTHESES ')' )
```

- The following restrictions apply only to ++HOLD MCSs packaged within a SYSMOD:
  - The only HOLD type allowed is SYSTEM.
  - ++HOLD statements must be placed after all ++VER and ++IF statements and before the first ++JCLIN or element statement.
  - The SYSMOD ID on the ++HOLD MCS must match the SYSMOD ID containing the ++HOLD MCS; that is, a SYSMOD cannot put another SYSMOD into exception SYSMOD status.

## Examples

The following examples are provided to help you use the ++HOLD MCS:

- “Example 1: Noting a Special Documentation Change”
- “Example 2: Marking a PTF That Is in Error” on page 544
- “Example 3: Specifying a Hold Class” on page 544

### Example 1: Noting a Special Documentation Change

Here is an example of a SYSMOD containing a ++HOLD statement. PTF UZ12345, which is applicable to function FXY1040, introduces a documentation change that the system operator should know about.

```

++PTF(UZ12345)          /* PTF modification.      */.
++VER(Z038) FMID(FXY1040) /* For MVS product FXY1040. */.
                        /* No prerequisites.      */.
++HOLD (UZ12345)        /* Hold this PTF          */.
      FMID(FXY1040)    /* for this function      */.
      SYSTEM           /* for system processing and*/.
      REASON(DOC)      /* for doc change.       */.
      COMMENT(this PTF changes operator console
                message xxx1233. A reply of U is
                required to this message ).
++MOD(IFBMOD01)        /* Change this module     */.
      DISTLIB(AOS12)   /* in this DLIB.         */.

```

### Example 2: Marking a PTF That Is in Error

When IBM discovers errors in PTFs it has already shipped, it provides information about these program error PTFs (PE-PTFs) in RETAIN. To make sure these PTFs are held, you can use the RETAIN information to build ++HOLD statements for the PTFs.

**Note:** In this example, because only ++HOLD statements are being shown, none of them are underlined.

```

++HOLD (UZ12345)        /* Hold this PTF          */.
      FMID(FXY1040)    /* for this function      */.
      ERROR            /* for APAR fix.         */.
      REASON(AZ00001)  /* APAR is AZ00001.     */.
      COMMENT(this APAR causes loop) /* */.
++HOLD (UZ12345)        /* Hold this PTF          */.
      FMID(FXY1040)    /* for this function      */.
      ERROR            /* for APAR fix.         */.
      REASON(AZ00002)  /* APAR is AZ00002.     */.
      COMMENT(this APAR causes 0C4) /* */.
++HOLD (UZ12346)        /* Hold this PTF          */.
      FMID(FXY1040)    /* for this function      */.
      ERROR            /* for APAR fix.         */.
      REASON(AZ00003)  /* APAR is AZ00003.     */.
      COMMENT(incorrect output on console) /* */.
++HOLD (UZ12347)        /* Hold this PTF          */.
      FMID(FXY1040)    /* for this function      */.
      ERROR            /* for APAR fix.         */.
      REASON(AZ00004)  /* APAR is AZ00004.     */.
                        /* Integrity - no comment. */.

```

### Example 3: Specifying a Hold Class

Here is an example of a SYSMOD containing a ++HOLD statement for a PTF requiring a UCLIN change unless a system generation is done:

```
++PTF(UZ12345)          /* PTF modification. */.  
++VER(Z038) FMID(FXY1040) /* For MVS product FXY1040. */  
                        /* No prerequisites. */  
++HOLD (UZ12345)        /* Hold this PTF */  
      FMID(FXY1040)    /* for this function */  
      SYSTEM           /* for system processing and*/  
      REASON(UCLIN)    /* for UCLIN change. */  
      COMMENT(UCLIN needed to change lmod for  
              IFBMOD01 to xxxxxxxx)  
      CLASS(SYSGEN)    /* The UCLIN change is not */  
                      /* needed if SYSGEN is done.*/.  
++MOD(IFBMOD01)        /* Change this module */  
      DISTLIB(AOS12)   /* in this DLIB. */.
```

## ++IF MCS

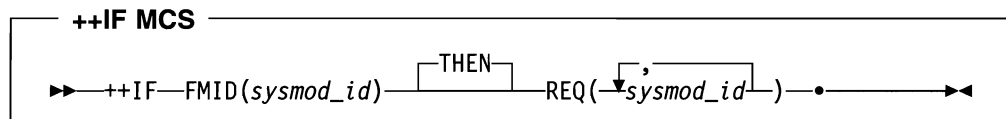
The ++IF MCS specifies requisites that must be installed with a SYSMOD if a certain function is installed. If the specified function is installed in the zone that SMP/E is trying to install this SYSMOD into, the requisite must also be installed in that zone; otherwise SMP/E does not install the SYSMOD containing the ++IF MCS. If the specified function is not yet installed in the zone where SMP/E is trying to install this SYSMOD, SMP/E saves the information from the ++IF MCS in case the specified function is installed later.

++IF statements are interpreted, reformatted, and placed in the target zone data set during APPLY processing and in the distribution zone data set during ACCEPT processing.

++IF statements can describe dependencies between two products, even if those products have different SRELs and are defined in different zones. The information from these statements is used by the REPORT command to identify cross-zone requisites—that is, conditional requisites needed in one zone because of ++IF statements in a SYSMOD installed in another zone.

A ++IF statement is associated with the ++VER MCS preceding it in the SYSMOD. Multiple ++IF statements can be specified following each ++VER MCS.

## Syntax



## Operands

### FMID

specifies the function that is to be checked to determine whether it is installed in one of the following:

- The target libraries (for APPLY processing)
- The distribution libraries (for ACCEPT processing)

This operand is required. It is not satisfied by superseded FMIDs.

### REQ

specifies the SYSMODs that are needed if the function SYSMOD specified on the FMID operand of the ++IF MCS is installed. This operand is required.

### THEN

specifies that the requisite for the ++IF MCS follows.

## Usage Notes

- The operands must be specified in the order shown in the syntax.
- The function specified for FMID must be different from the SYSMOD ID on the header statement, the FMID operand value on the header statement, and the FMID operand value on the ++VER MCS associated with this ++IF MCS.

## Example: PTF with ++IF

PTF UZ00004 contains service for elements belonging to function FXY1040. If function FXY1050 is installed in the same zone, PTF UZ00005 must also be installed in that zone in order for PTF UZ00004 to be installed successfully. If function FXY1050 is not presently installed, PTF UZ00005 is not required; however, SMP/E saves the ++IF MCS in case function FXY1050 is installed in the future. Then, when FXY1050 is installed, PTF UZ00005 is considered an unsatisfied conditional requisite that must also be installed.

Here is an example of a SYSMOD containing a ++IF statement to define UZ00005 as a conditional requisite:

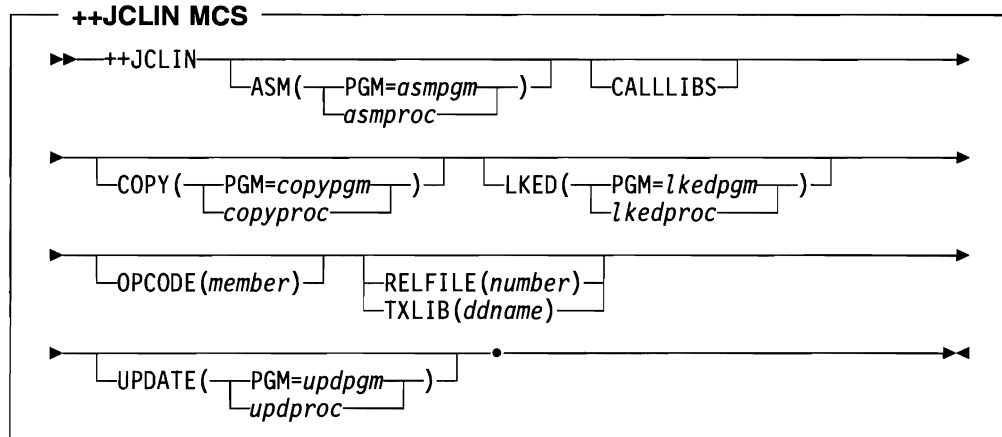
```
++PTF(UZ00004)          /* PTF SYSMOD.          */.  
++VER(Z038) FMID(FXY1040) /* For MVS product FXY1040. */.  
++IF FMID(FXY1050) /* If FXY1050 is on, or */  
    THEN /* when it's put on later, */  
    REQ(UZ00005) /* UZ00005 is required. */.  
++MOD(IFBMOD01) /* Replace this module */  
    DISTLIB(AOSFB) /* in this DLIB. */.  
++MACUPD(IFBMAC01) /* Update this macro */  
    DISTLIB(IFBMACS) /* in this DLIB. */.
```

## ++JCLIN MCS

The ++JCLIN MCS identifies the start of the job control language (JCL) input that must be processed as part of installing the SYSMOD. The JCLIN input is used to identify the structure of the target system libraries to SMP/E so that the modules, macros, and source code being replaced can be installed correctly. Processing of ++JCLIN input during APPLY or ACCEPT is similar to the processing done for the JCLIN command. For an additional description of JCLIN processing, see Chapter 8.

**Note:** The ++JCLIN MCS does **not** cause SMP/E to update the target or distribution libraries; only the entries in the target and distribution zones are updated. These libraries are updated when SMP/E processes the elements in the SYSMOD. The element statements in the SYSMOD determine which elements should be installed and cause SMP/E to invoke the utilities that install the elements.

## Syntax



## Operands

### ASM

specifies the name of the assembler program or procedure that is used in the JCLIN data. This operand must be specified if the name is different from those recognized by SMP/E, which are the program names ASMBLR, ASMA90, IEUASM, IEV90, and IFOX00, and procedure name ASMS.

### CALLLIBS

specifies that SMP/E is to process SYSLIB DD statements in JCLIN link-edit steps. If the CALLLIBS operand is not specified on either the JCLIN command or the ++JCLIN statement, SMP/E JCLIN processing ignores any SYSLIB DD statements that it encounters.

### COPY

specifies the name of the copy program or procedure that is used in the JCLIN data. This operand must be specified if the name is different from that recognized by SMP/E, which is the program name IEBCOPY.

### LKED

specifies the name of the link-edit program or procedure that is used in the JCLIN data. This operand must be specified if the name is different from those



recognized by SMP/E, which are the program names HEWL, IEWL, and IEWBLINK, and procedure name LINKS.

**OPCODE**

identifies a PARMLIB member containing control statements identifying character strings to be treated as opcodes or macros.

**PGM**

specifies a program name (instead of a procedure name) for a utility.

**RELFIL**

specifies the relative position, within the files associated with this SYSMOD, of the file containing the JCLIN data as one of its members.

The member in the RELFILE data set containing the JCLIN input data must match the SYSMOD's SYSMOD ID; for example, the JCLIN for ++PTF(UZ11111) is supplied in member UZ11111 of the unloaded PDS identified by the RELFILE operand.

**Note:** RELFILE is mutually exclusive with TXLIB.

**TXLIB**

specifies the ddname of a partitioned data set containing the JCLIN data as one of its members.

The member of the TXLIB data set containing the JCLIN input data must match the SYSMOD's SYSMOD ID; for example, the JCLIN for ++PTF(UZ11111) is supplied in member UZ11111 of the data set identified by the TXLIB operand.

**Notes:**

1. SMPTLIB cannot be used as a value on the TXLIB operand.
2. TXLIB is mutually exclusive with RELFILE.

**UPDATE**

specifies the name of the update program or procedure that is used in the JCLIN data. This operand must be specified if the name is different from that recognized by SMP/E, which is the program name IEBUPDTE.

## Usage Notes

- If the JCLIN data is packaged inline, it must immediately follow the ++JCLIN MCS and must not contain any records that start with ++. Neither **RELFIL** nor **TXLIB** can be specified on the ++JCLIN MCS.
- If the JCLIN data is packaged in a TXLIB data set, the ddname specified on the TXLIB operand is required during APPLY and ACCEPT processing.
- For information about packaging SYSMODs in RELFILE format, see the *Standard Packaging Rules for MVS-Based Products* manual.

## Examples

For these examples, assume function JXY1040 contains some inline JCLIN, one module (IFBMOD01), and one macro (IFBMAC01). The same format was used to package the JCLIN and the elements.

**Note:** Even though the examples are for a function, they are equally valid for PTFs, APARs, and USERMODs.

The following examples are provided to help you use the ++JCLIN MCS:

- “Example 1: ++JCLIN Data Packaged Inline”
- “Example 2: ++JCLIN Data Packaged in a RELFILE”
- “Example 3: ++JCLIN Data Packaged in a TXLIB with a User Utility Program Name” on page 551

### Example 1: ++JCLIN Data Packaged Inline

Here is an example of a SYSMOD containing a ++JCLIN statement for packaging JCLIN inline:

```

++FUNCTION(JXY1040)      /* Function SYSMOD.          */.
++VER(Z038)             /* For MVS, no requisites. */.
++JCLIN                 /* JCLIN is required.      */.
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1    EXEC PGM=IEWL
//SYSLMOD  DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12    DD DSN=SYS1.AOS12,DISP=SHR
//SYSLIN   DD *
            INCLUDE AOS12(IFBMOD01)
            ENTRY  IFBMOD01
            ALIAS  IFBMOD01A
            NAME   IFBMOD01(R)
/*
++MOD(IFBMOD01)         /* This module            */.
                        DISTLIB(AOS12) /* for this DLIB.        */.
...
... object deck for IFBMOD01
...
++MAC(IFBMAC01)        /* This macro             */.
                        DISTLIB(AMACLIB) /* for this DLIB.       */.
...
... macro replacement for IFBMAC01
...

```

No special DD statements, other than those for the target and DLIBs, are required to process this SYSMOD.

During APPLY, SMP/E processes the ++JCLIN data, determines that module IFBMOD01 gets linked into load module IFBMOD01 (which has an alias), stores information in the target zone, and then invokes the proper utilities to install the module and macro.

### Example 2: ++JCLIN Data Packaged in a RELFILE

Here is an example of a SYSMOD containing a ++JCLIN statement for packaging JCLIN in relative files. No special ++JCLIN operands are required.

```

++FUNCTION(JXY1040)      /* Function SYSMOD      */
      FILES(3)           /* in RELFILE format.  */
++VER(Z038)             /* For MVS, no requisites. */
++JCLIN                  /* JCLIN is required.  */
      RELFILE(1)         /* Data is in first RELFILE.*/
++MOD(IFBMOD01)         /* This module          */
      DISTLIB(AOS12)     /* for this DLIB        */
      RELFILE(2)         /* is in second RELFILE. */
++MAC(IFBMAC01)         /* This macro           */
      DISTLIB(AMACLIB) /* for this DLIB        */
      RELFILE(3)         /* is in third RELFILE.  */

```

The SMPTLIB DD statement is required for the RECEIVE command to load the RELFILE data sets from tape onto DASD, and at APPLY and ACCEPT, in order to access the data in the unloaded partitioned data sets on DASD.

**Note:** Different RELFILE data sets are required for the module and macro, because they are in different formats (modules must be in load module format, RECFM=U, while macros are RECFM=FB). Although a separate RELFILE data set was used to contain the ++JCLIN data, that data could have been put into the data set with the macros, because both have the same attributes (LRECL=80, RECFM=FB).

### Example 3: ++JCLIN Data Packaged in a TXLIB with a User Utility Program Name

Here is an example of a SYSMOD containing a ++JCLIN statement for packaging JCLIN in text libraries. The JCLIN contains assembly steps using a special link-edit utility, ALTIEWL.

```

++FUNCTION(JXY1040)      /* Function SYSMOD.      */
++VER(Z038)             /* For MVS, no requisites. */
++JCLIN                  /* JCLIN is required.  */
      TXLIB(LIB1040)     /* Data is in text library. */
      LKED(PGM=ALTIEWL) /* Use other link-edit.  */
++MOD(IFBMOD01)         /* This module          */
      DISTLIB(AOS12)     /* for this DLIB        */
      TXLIB(LIB1040)     /* is in text library.  */
++MAC(IFBMAC01)         /* This macro           */
      DISTLIB(AMACLIB) /* for this DLIB        */
      TXLIB(LIB1040)     /* is in text library.  */

```

A DD statement for LIB0104 is required during APPLY and ACCEPT in order for SMP/E to call the utilities to get the replacements installed into the operating system libraries. An example of the DD statement is:

```
//LIB1040 DD DSN=...
```

**Note:** In this case, the replacements for all three elements were put in one data set. That data set must contain three members: member JXY1040 must be the JCLIN data, member IFBMOD01 must contain the object deck for module IFBMOD01, and member IFBMAC01 must contain the replacement for macro IFBMAC01. Because all have the same formats (that is, RECFM=FB LRECL=80) they can be packaged in one data set.

Member JXY1040 looks as follows:

```
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1    EXEC PGM=ALTIEWL
//SYSLMOD  DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12    DD DSN=SYS1.AOS12,DISP=SHR
//SYSLIN   DD *
           INCLUDE AOS12(IFBMOD01)
           ENTRY  IFBMOD01
           ALIAS  IFBMOD01A
           NAME  IFBMOD01(R)
/*
```

Note the use of ALTIEWL on the EXEC statement.

## ++MAC MCS

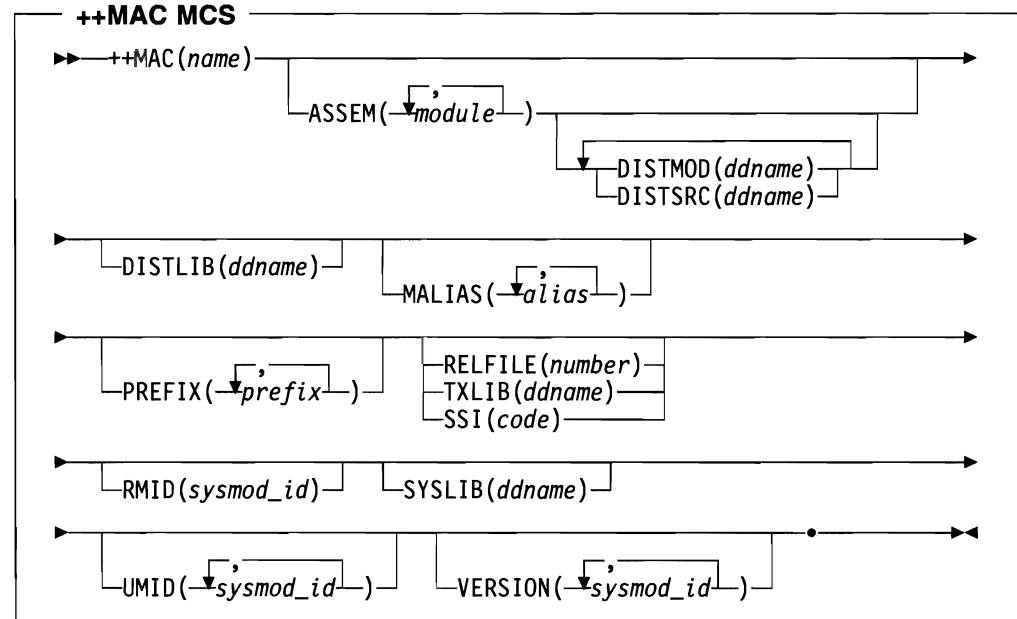
The ++MAC MCS describes a single macro replacement. It must immediately precede the macro definition statements when they are within the SYSMOD.

## Syntax

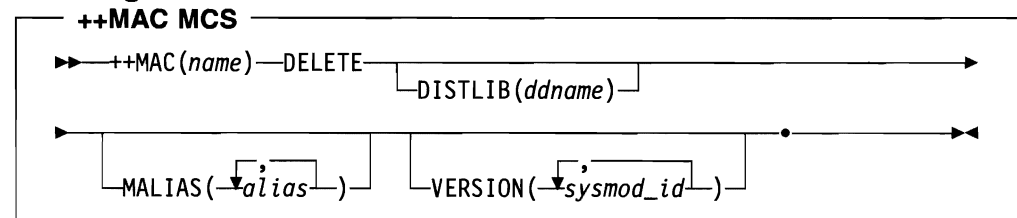
The syntax to be used depends on the processing to be done for the element:

- Adding or replacing the element
- Deleting the element

### Adding or Replacing a Macro



### Deleting a Macro



## Operands

### ASSEM

specifies the names of modules to be assembled in addition to those modules named as GENASM subentries in the MAC entry. The source for the assemblies is the ASSEM entry, SRC entry, or DISTSRC member whose name matches the specified ASSEM value. SMP/E looks for a match—first with an ASSEM entry, then with a SRC entry, and finally with an entry in the DISTSRC data set—and uses the first match it finds. The source must either be known to SMP/E at the time the ASSEM operand is encountered on the ++MAC statement, or be included in the same SYSMOD.

**Notes:**

1. APPLY and ACCEPT processing place the specified names into the SYSMOD entry created on the target zone and distribution zone.
2. If the object deck for the element specified on the ASSEM operand is also provided by the SYSMOD, the assembly may not occur. (See "Assemblies" on page 95 for more information.)

**DELETE**

specifies that the macro is to be removed from the target libraries, distribution libraries, and SMP/E data sets.

**Notes:**

1. DELETE is mutually exclusive with all other operands except DISTLIB, MALIAS, and VERSION.
2. If the element statement is in a base function, you may want to use the DELETE operand on the ++VER MCS to delete the previous release, rather than on the element statement to delete a specific element.

**DISTLIB**

specifies the ddname of the distribution library for the specified macro.

**Notes:**

1. **DISTLIB** must be specified if the macro has not been previously recorded on the target zone or distribution zone. If a MAC entry already exists in the target zone or distribution zone and the value currently in that entry does not match that specified in the DISTLIB operand, the SYSMOD is not applied or accepted, unless that SYSMOD also used the ++MOVE MCS to change the DISTLIB to that new value.
2. You cannot use SYSPUNCH as the DISTLIB. It is used by SMP/E and other products to process assembled modules.

**DISTMOD**

specifies the ddname of the link-edit distribution library for those modules specified in the ASSEM operand. During ACCEPT processing, the object code from the assembler is link-edited to the library specified.

**DISTSRC**

specifies the ddname of the library containing the additional assembly or source to be assembled. The additional assembly or source must be specified in the ASSEM operand.

**MALIAS**

specifies the alias names for the macro in both the target system and the distribution libraries.

You can use MALIAS when two or more macros that must be defined in the same zone must have the same name for programming access. For example, you can use MALIAS if several products have a help macro whose name must match the name of the command processing module it describes. You can specify HELP on MALIAS and a unique element name as the macro name.

*name*

specifies the name of the macro member in the distribution library and, optionally, in the target system library. The name can contain any alphanumeric characters and \$, #, @, or hex C0.

**PREFIX**

specifies the first characters (prefix) of the names of modules to be assembled in addition to those modules named as GENASM subentries in the target zone MACRO entry. The prefix values must contain no more than 7 characters.

The full module names are determined by comparing the prefix with the target zone or distribution zone MOD entry names.

The source for the assembly is the ASSEM entry, SRC entry, or DISTSRC member whose name matches a MOD entry name beginning with one of the specified prefixes. SMP/E looks for a match—first with an ASSEM entry, then with a SRC entry, finally with an entry in the DISTSRC data set—and uses the first match it finds. The source must either be known to SMP/E at the time the PREFIX operand is encountered on the ++MAC statement, or be included in the same SYSMOD.

**Note:** If the object deck for an element selected by the PREFIX operand is also provided by the SYSMOD, the assembly may not occur. (See “Assemblies” on page 95 for more information.)

**RELFILE**

identifies which relative file associated with the SYSMOD contains this element. This operand is required if you provide the element in RELFILE format, rather than inline or in a TXLIB data set.

**Note:** RELFILE is mutually exclusive with TXLIB.

**RMID**

specifies the last SYSMOD that **replaced** this macro. This operand may be used only in a service-updated function, and the specified PTF must be integrated into the function.

**SSI**

specifies eight hexadecimal digits of system status information. This information is placed in the directory of the target system library or SMPMTS or SMPSTS during APPLY processing, and in the distribution library during ACCEPT processing, as four packed hexadecimal bytes of user data. See the IEBUPDTE program description in the utilities manual for your operating system.

**Note:** This operand is ignored if text is located in a library, as is the case when either the RELFILE or TXLIB operand is specified.

**SYSLIB**

specifies the ddname of the target library, if the macro exists in one. APPLY and RESTORE processing update this library.

**Note:** If a MAC entry already exists in the target zone or distribution zone and the value currently in that entry does not match that specified in the SYSLIB operand, SMP/E ignores the SYSLIB value in the SYSMOD being installed, unless that SYSMOD also used the ++MOVE MCS to change the SYSLIB to that new value.

**TXLIB**

is the ddname of the partitioned data set containing the macro. This operand is required if the macro is provided in a data set that the users have access to, rather than inline or in RELFILE format.

**Notes:**

1. SMPTLIB cannot be used as a value on the TXLIB operand.
2. TXLIB is mutually exclusive with RELFILE.

**UMID**

specifies the SYSMODs that have **updated** this macro since it was last replaced. This operand can be used only in a service-updated function, and the specified PTFs must be integrated into the function.

**VERSION**

specifies one or more function SYSMODs that currently contain the element. The function containing the ++MAC MCS will take over ownership of the element from the specified functions.

When **VERSION** is specified on an element statement, it overrides any VERSION operand values that might be specified on the ++VER MCS.

## Usage Notes

- If the macro is packaged inline, it must immediately follow the ++MAC MCS and must not contain any records that start with ++. Neither RELFILE nor TXLIB can be specified on the ++MAC MCS.
- If the macro is packaged in a TXLIB data set, the ddname specified in the TXLIB operand is required during APPLY and ACCEPT processing.
- For information about elements packaged in RELFILE format, see the *Standard Packaging Rules for MVS-Based Products* manual.
- If the ++MAC MCS is for an inner macro (a macro referred to by another macro instruction that resides in the macro library), the modules that require reassembly must be specified in the ASSEM operand list.
- If the macro resides in a target library (rather than the SMPMTS), the target library should be included in the SYSLIB DD concatenation for assemblies during APPLY processing. For additional information on SYSLIB requirements, see “Usage Notes” on page 67 and on pages 19 and 336, and Appendix B.

## Examples

The following examples are provided to help you use the ++MAC MCS:

- “Example 1: Replacing a Macro through a USERMOD”
- “Example 2: Deleting a Macro” on page 557
- “Example 3: Adding a New Macro” on page 557

**Example 1: Replacing a Macro through a USERMOD**

Assume you want to replace a macro that is part of an IBM product, such as GIMOPCDE, which is supplied with SMP/E. You have updated a copy of the macro in a text library, TSO.NEWSMP.MACLIB. Here is an example of a SYSMOD containing a ++MAC statement to build a USERMOD that installs the changes:



```

++USERMOD(USR0001)      /* User modification      */.
++VER(Z038) FMID(HMP1802) /* for SMP/E              */.
++MAC(GIMOPCDE)         /* to replace this macro. */
      TXLIB(NEWSMP) /* Macro is in this TXLIB. */.

```

In this example, you have just applied another piece of service to the SMP/E macro, but SMP/E is still the owner of the macro. If you attempt to install some IBM-supplied service (that is, a PTF) to that macro, SMP/E issues an error message indicating that your user modification will be regressed, and will not install that service until the BYPASS(ID) operand is used.

Another method of installing the new macro is for you to assume ownership for the macro by using the VERSION operand. Assume you already have a user function, JXY1040, installed, and you want to transfer ownership of the SMP/E macro to your function. The following SYSMOD contains a ++MAC statement to do that:

```

++USERMOD(USR0001)      /* User modification      */.
++VER(Z038) FMID(JXY1040) /* for user application   */.
++MAC(GIMOPCDE)         /* to replace this macro. */
      VERSION(HMP1802) /* Version SMP/E.        */.
      TXLIB(NEWSMP) /* Macro is in this TXLIB. */.

```

If after the installation of this SYSMOD any subsequent IBM service modifies this macro, the replacement or update from the IBM service is not selected. It is your responsibility to provide continued modifications for the macro. Thus, this method of updating an element should be used carefully.

In both examples, because the new macro exists in a TXLIB, the following DD statement is required during APPLY and ACCEPT:

```
//NEWSMP DD DSN=TSO.NEWSMP.MACLIB,DISP=SHR
```

### Example 2: Deleting a Macro

Assume you have installed one of your application programs as function HUSR001. The function contains macro USRMAC01, which is no longer required. Here is an example of a SYSMOD containing a ++MAC statement to delete the macro from the target and distribution libraries:

```

++USERMOD(USR0001)      /* User modification      */.
++VER(Z038) FMID(HUSR001) /* for user application.   */.
++MAC(USRMAC01)         /* Delete this macro.     */
      DELETE /*

```

### Example 3: Adding a New Macro

Assume you have installed one of your application programs as function HUSR001. A new macro USRMAC02 is required, and it has these requirements:

- The macro is to be put into SYS1.USRMACS in the target zone.
- The macro is to be put into SYS1.AUSRMACS in the distribution zone.
- The macro is to be installed with an alias of TERMINAL.
- After being installed, modules USRASM01, USRASM02, USRSRC01, and USRSRC02 are to be assembled. USRASM01 and USRASM02 already exist as assembler entries in the CSI, and USRSRC01 and USRSRC02 exist as source entries in the CSI.

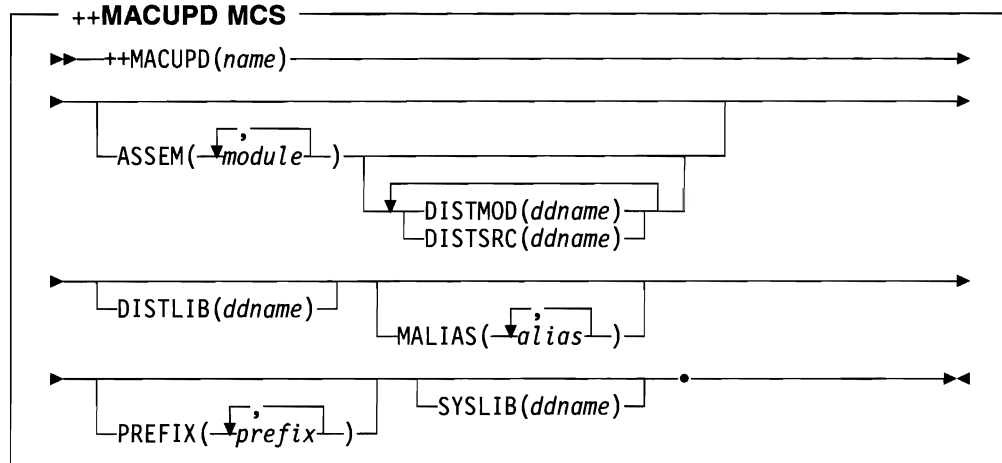
Here is an example of a SYSMOD containing a ++MAC statement defining all this information:

```
++USERMOD(USR0002)      /* User modification      */.  
++VER(Z038) FMID(HUSR001) /* for user application. */.  
++MAC(USRMAC02)        /* Add this macro      */.  
    DISTLIB(AUSRMACS) /* to this DLIB      */  
    SYSLIB(USRMACS) /* and to this tgt lib  */  
    MALIAS(TERMINAL)/* with this alias.    */  
    ASSEM(USRASM01, /* Assemble these modules. */  
        USRASM02, /* */  
        USRSRC01, /* */  
        USRSRC02) /* */  
        /* Inline text follows. */.  
  
...  
... macro USRMAC02 goes here  
...
```

## ++MACUPD MCS

The ++MACUPD MCS describes a single macro update within a PTF, an APAR fix, or a USERMOD. It must immediately precede the macro update statements within the SYSMOD.

### Syntax



### Operands

#### ASSEM

specifies the names of modules to be assembled in addition to those named as GENASM subentries in the MAC entry. The source for the assemblies is the ASSEM entry, SRC entry, or DISTSRC member whose name matches the specified ASSEM value. SMP/E looks for a match—first with an ASSEM entry, then with a SRC entry, and finally with an entry in the DISTSRC data set—and uses the first match it finds. The source must either be known to SMP/E at the time the ASSEM operand is encountered on the ++MACUPD statement, or be included in the same SYSMOD.

#### Notes:

1. APPLY and ACCEPT processing place the specified names into the SYSMOD entry created on the target zone and distribution zone.
2. If the object deck for the element specified on the ASSEM operand is also provided by the SYSMOD, the assembly may not occur. For more information, see “Assemblies” on page 95.

#### DISTLIB

specifies the ddname of the distribution library for the specified macro.

#### Notes:

1. **DISTLIB** must be specified if the macro has not been previously recorded on the target zone or distribution zone. If a MAC entry already exists in the target zone or distribution zone, and the value currently in that entry does not match that specified in the DISTLIB operand, the SYSMOD is not applied or accepted.
2. You cannot use SYSPUNCH as the DISTLIB. It is used by SMP/E and other products to process assembled modules.

**DISTMOD**

specifies the ddname of the link-edit distribution library for the modules specified in the ASSEM operand. During ACCEPT processing, the object code from the assembler is link-edited to the library specified.

**DISTSRC**

specifies the ddname of the library containing the additional assembly or source to be assembled. The additional assembly or source must be specified in the ASSEM operand.

**MALIAS**

specifies the alias names for the macro in both the target system and distribution libraries.

You can use MALIAS when two or more macros that must be defined in the same zone must have the same name for programming access. For example, you can use MALIAS if several products have a help macro whose name must match the name of the command processing module it describes. You can specify HELP on MALIAS and a unique element name as the macro name.

*name*

specifies the name of the macro member in the distribution library and, optionally, in the target system library. The name can contain any alphanumeric characters and \$, #, @, or hex C0.

**PREFIX**

specifies the first characters (prefix) of the names of modules to be assembled in addition to those modules named as GENASM subentries in the target zone MAC entry. The prefix values must be 7 characters or less.

The full module names are determined by comparing the prefix with the target zone or distribution zone MOD entry names.

The source for the assembly is the ASSEM entry, SRC entry, or DISTSRC member whose name matches a MOD entry name beginning with one of the specified prefixes. SMP/E looks for a match—first with an ASSEM entry, then with a SRC entry, and finally with an entry in the DISTSRC data set—and uses the first match it finds. The source must be either known to SMP/E at the time the PREFIX operand is encountered on the ++MACUPD statement, or be included in the same SYSMOD.

**Note:** If the object deck for the element specified on the PREFIX operand is also provided by the SYSMOD, the assembly may not occur. For more information, see “Assemblies” on page 95.

**SYSLIB**

specifies the ddname of the target library, if the macro exists in one. APPLY and RESTORE processing update this library.

**Usage Notes**

- | • If a SYSMOD containing a ++MACUPD statement attempts to change the ownership (FMID) of the element (via the VERSION operand), the SYSMOD cannot be installed.
- |
- | • The changes for the macro must immediately follow the ++MACUPD MCS and must not contain any records that start with ++.

- If the macro resides in a target library (rather than the SMPMTS), that target library should be included in the SYSLIB DD concatenation for assemblies during APPLY processing. For more information on SYSLIB requirements, see “Usage Notes” on page 67 and on pages 19 and 336, and Appendix B.
- The only IEBUPDTE control statements allowed in a SYSMOD are `./ CHANGE` and `./ ENDUP`.
- The only IEBUPDTE CHANGE operand SMP/E checks is NAME, which must specify the same element as the ++MACUPD MCS. Other CHANGE operands may produce undesired results and are used at your own risk. For example, if you code `UPDATE=INPLACE`, SMP/E may update the distribution library. Once the distribution libraries are changed, there is no way to remove the updates.
- SMP/E does not support a continuation of the `./ CHANGE` statement.
- SMP/E generates any `./ ALIAS` statements needed and places them in the IEBUPDTE input data following the last text statement. The `./ ALIAS` control statements are generated only for macro updates.
- When processing multiple updates to the same lines in a given macro, SMP/E uses the `./ CHANGE` statement from the last update to the lines.
- If an APAR fix or USERMOD updates a macro that causes an assembly, SMP/E sets the ASSEMBLY indicator in the MOD entry for the assembled module. This can cause a problem when additional service that does not know about the macro change is installed at the same time as the APAR or USERMOD—for example, if you are installing your own USERMOD and IBM-supplied PTFs with the same APPLY command. In such cases, because the ASSEMBLY indicator is set, the module is reassembled, but does not contain the macro changes from the APAR or USERMOD. To prevent these assemblies, you can reset the ASSEMBLY indicator using UCLIN after installing the APAR or USERMOD.

## Example: Updating a Macro That Causes an Assembly

Assume you want to update macro IFBMAC02, which resides in distribution library IFBMACS. Because of this change, module IFBSRC01 must be reassembled. Module IFBSRC01 exists as a source in distribution library SYS1.IFBSRC, and as an object module in distribution library SYS1.AOS23. The macro and the modules are part of JXY1040, a user-written function. Here is an example of a SYSMOD containing a ++MACUPD statement to make the necessary changes:

```

++PTF(USR0001)          /* Preventive service      */.
++VER(Z038) FMID(JXY1040) /* for user product.      */.
++MACUPD(IFBMAC02)      /* Update this macro      */.
                        DISTLIB(IFBMACS)/* in this DLIB.      */.
                        ASSEM(IFBSRC01) /* Assemble this source.  */.
                        DISTSRC(IFBSRC) /* Source is here.       */.
                        DISTMOD(AOS23) /* Assembled SRC goes here. */.
./ CHANGE name=IFBMAC02
... IEBUPDTE control cards and data
...

```

In this example, DD statements are required when the SYSMOD is applied to define the target libraries for the macro and the load module to be updated as a result of the assembly. For example, if the modules in SYS1.AOS23 (the assembled module's distribution library) were copied to SYS1.LINKLIB and the source in SYS1.IFBSRC (the source element's distribution library) were copied to SYS1.CHGLIB, the following DD statements are needed:

```

//LINKLIB DD DSN=SYS1.LINKLIB...
//CHGLIB  DD DSN=SYS1.CHGLIB...

```

In this example, the following DD statements are needed when the SYSMOD is being accepted to define the distribution libraries:

```

//IFBMACS DD DSN=SYS1.IFBMACS... (macro DLIB)
//IFBSRC  DD DSN=SYS1.IFBSRC... (source DLIB for assembly)
//AOS23   DD DSN=SYS1.AOS23... (DLIB for module assembled)

```

## ++MOD MCS

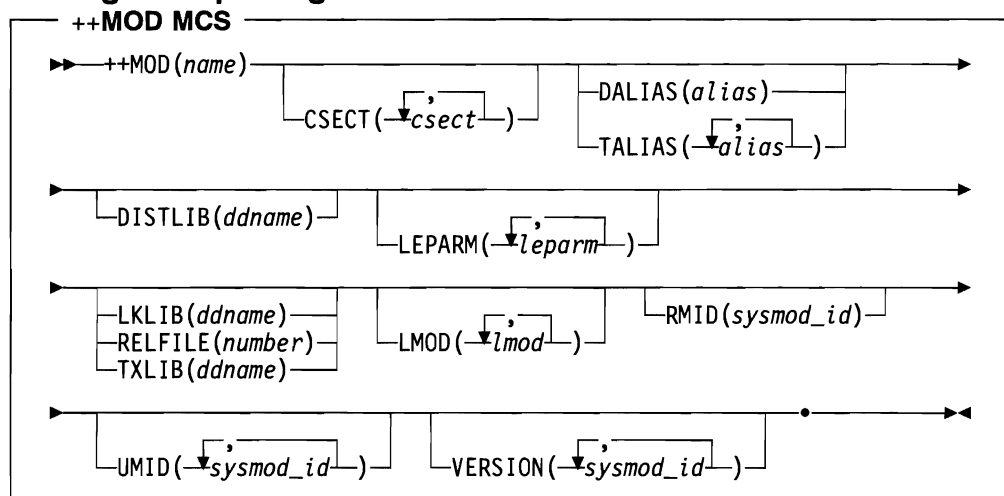
The ++MOD MCS describes a single module replacement. It must immediately precede the module definition statements when they are within the SYSMOD. You should use the ++MOD MCS when you want to provide the object form of a module. If you want to provide the source form and have it assembled when the SYSMOD is installed, use the ++SRC MCS instead.

## Syntax

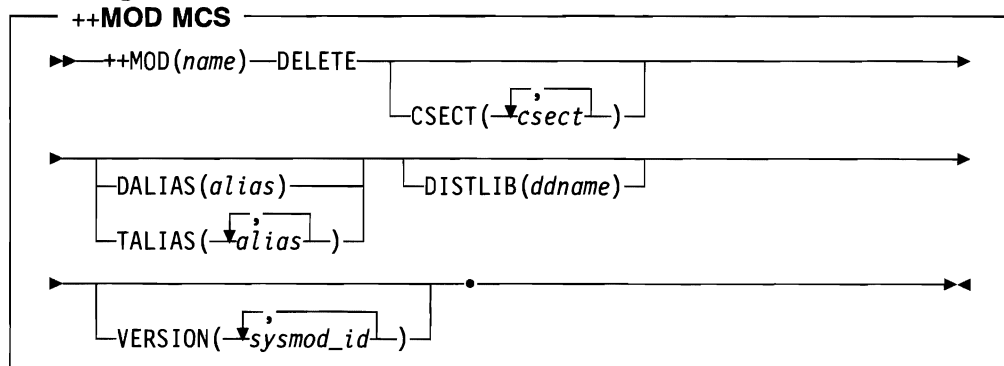
The syntax to be used depends on the processing to be done for the element:

- Adding or replacing the element
- Deleting the element

### Adding or Replacing a Module



### Deleting a Module



## Operands

### CSECT

lists all the CSECTs contained in the module. This operand is required if the module contains more than one CSECT or if the CSECT name is different from the module name on the ++MOD MCS.

- If no CSECT name is specified, SMP/E assumes that the module contains only one CSECT, whose name matches the module name on the ++MOD MCS.
- If CSECT is specified, it must include all the CSECTs contained in the module, even if one of them has the same name as the module.

**Notes:**

1. A CSECT name can contain from 1 to 8 characters. The name can contain any characters except the following:

- Comma ,
- Left parenthesis (
- Right parenthesis )
- Blank

2. Comments are not allowed within a CSECT name. For example, the following is not allowed:

```
CSECT ( /* this is a csect name */ CSECT01)
```

The comment is interpreted as part of the CSECT name, instead of a comment.

3. Even if CSECT is not specified on the ++MOD MCS used to create a MOD entry, CSECT information is saved if CSECT is specified on subsequent ++MOD statements that update the MOD entry.

**DALIAS**

is the alias name of a module that has an alias in the distribution library, but not in the target library. This might be used if the module is included under its alias name during system generation.

**Note:** DALIAS is mutually exclusive with TALIAS.

**DELETE**

indicates that the module is to be removed from the target libraries, distribution libraries, and SMP/E data sets.

**Note:** DELETE is mutually exclusive with all other operands except CSECT, DALIAS, DISTLIB, TALIAS, and VERSION.

**DISTLIB**

specifies the ddname of the distribution library for the specified module.

**Notes:**

1. This operand must be specified if the module has not been previously recorded on the target zone or distribution zone. If a MOD entry already exists in the target zone or distribution zone and the value currently in that entry does not match that specified in the DISTLIB operand, the SYSMOD is not applied or accepted, unless that SYSMOD also used the ++MOVE MCS to change the DISTLIB to that new value.
2. You cannot use SYSPUNCH as the DISTLIB. It is used by SMP/E and other products to process assembled modules.



**LEPARM**

specifies link-edit utility attributes for the module. Any of the following values can be specified:

AC=1	AMODE=MIN	OL	RMOD=31
ALIGN2	AMOD=MIN	OVLY	RMODE=ANY
AMODE=24	DC	REFR	RMOD=ANY
AMOD=24	FETCHOPT(PACKINOPACK,PRIMEINOPRIME)		
AMODE=31	MAXBLK( <i>nnnnn</i> )	RENT	SCTR
AMOD=31	NE	REUS	STD
AMODE=ANY	NOCALL	RMODE=31	
AMOD=ANY	NCAL		

**Notes:**

1. The LEPARM values from the ++MOD MCS are associated with a load module entry only if the module was copied, not link-edited, into the target libraries. (The COPY indicator is set in the load module entry.) If the load module was link-edited, JCLIN must be used to change its link-edit utility attributes.
2. During APPLY processing of a ++MOD MCS with LEPARMS, the LEPARM options are saved not in the MOD entry created, but in the LMOD entry. During ACCEPT processing, the MOD entry is created with the LEPARMS present. The target zone MOD entry can contain the LEPARM options through either UCLIN or the copying of the distribution zone to a target zone.
3. The link-edit attributes listed above are the only attributes that can be specified on the LEPARM operand. If any other attributes are specified, a syntax error will result during RECEIVE processing.

For more information on how the LEPARM operand is processed, see "Load Module Attributes and Link-Edit Parameters" on page 46 in the chapter on ACCEPT, and on page 97 in the chapter on APPLY. These attributes are described in full in "MOD Entry (Distribution and Target Zone)" on page 723.

**LKLIB**

is the ddname of the partitioned data set containing the link-edited format of the object module. This operand is required if the module is provided in a data set, rather than inline or on a tape.

**Note:** LKLIB is mutually exclusive with RELFILE and TXLIB.

**LMOD**

lists existing load modules that are to contain the module. If any of the names specified are not already LMOD subentries in the target zone MOD entry, they are added during APPLY processing.

**Notes:**

1. LMOD can be used only to add a module to an existing load module.
2. LMOD cannot be used to create a new load module. Nor can it be used if any link-edit control statements must be added or changed to add the module to an existing load module. However, you can use JCLIN data to create a new load module and to add or change link-edit control statements.

3. If an LMOD entry does not exist for one of the load modules specified, sufficient information is not available to create one. Thus, when the MOD is to be link-edited during APPLY processing, a warning message is issued, and no link-edit is performed for that load module.

*name*

specifies the name of the module in the distribution library and, optionally, in the target library. The name can contain any alphanumeric characters and \$, #, @, or hex C0.

**RELFILE**

identifies which relative file associated with the SYSMOD contains this module. This operand is required if you provide the element in RELFILE format, rather than inline or in a LKLIB or TXLIB data set.

**Notes:**

1. RELFILE is mutually exclusive with LKLIB and TXLIB.
2. If an object module is provided in RELFILE format, it must be in link-edited format.

**RMID**

specifies the last SYSMOD that **replaced** this module. This operand can be used only in a service-updated function, and the specified PTF must be integrated into the function.

**TALIAS**

specifies one or more alias names for the module. The aliases exist in the distribution library and the target library.

You can use TALIAS for a module that was copied from a distribution library into a target library (defined by JCLIN data as a copied module), but not for one that is link-edited (defined by JCLIN data as a link-edited module). **TALIAS** must be specified on the ++MOD MCS even if **ALIAS** was specified on the COPY SELECT statement.

Likewise, to specify an alias for a copied load module, you must use the TALIAS operand on the ++MOD statement for that load module. (To specify an alias for a link-edited load module, do not use TALIAS. You must identify that alias using an ALIAS link-edit control statement in the JCLIN that defined the load module. For more information, see "Processing Link-Edit Steps" on page 188.)

**Note:** TALIAS is mutually exclusive with DALIAS.

**TXLIB**

is the ddname of the partitioned data set containing an object module that has not been link-edited. This operand is required if the module is provided in a TXLIB data set rather than inline, in a LKLIB data set, or in RELFILE format.

**Notes:**

1. SMPTLIB cannot be used as a value on the TXLIB operand.
2. TXLIB is mutually exclusive with LKLIB and RELFILE.

**UMID**

specifies the SYSMODs that have **updated** this module since it was last replaced. This operand can be used only in a service-updated function, and the specified PTFs must be integrated into the function.

**VERSION**

specifies one or more function SYSMODs that currently contain the element. The function containing the ++MOD MCS takes over ownership of the element from the specified functions.

When VERSION is specified on an element statement, it overrides any VERSION operand values specified on the ++VER MCS.

**Usage Notes**

- If the module is packaged inline, it must immediately follow the ++MOD MCS and must not contain any records that start with ++. Neither **LKLIB**, **RELFILE**, nor **TXLIB** may be specified.
- If the module is packaged in a TXLIB data set, the ddname specified on the TXLIB operand is required during APPLY and ACCEPT processing.
- If the module is in an LKLIB data set, the ddname specified in the LKLIB operand is required during APPLY and ACCEPT processing. Module replacements in an LKLIB data set must be in load module format (that is, processed by the link-edit utility).
- For information about packaging SYSMODs in RELFILE, TXLIB, or inline format, see the *Standard Packaging Rules for MVS-Based Products* manual.
- There are several ways to associate a module with a load module:
  - The DISTLIB library can be totally copied into the target library.
  - JCLIN can identify the module as part of one or more load modules.
  - The LMOD operand on the ++MOD MCS can indicate the associated load module.

If SMP/E cannot identify the load module associated with a given module, it does not update the target libraries during APPLY processing. Instead, it issues warning message GIM43401.

**Examples**

The following examples are provided to help you use the ++MOD MCS:

- “Example 1: Adding a New Module to an Existing Load Module”
- “Example 2: Specifying Link-Edit Utility Attributes with LEARM” on page 568

**Example 1: Adding a New Module to an Existing Load Module**

Module IFBMOD01 is a new module that is to be placed in the distribution library SYS1.AOSFB and is to be link-edited with the existing load module IEEFRQ in the target system library SYS1.LINKLIB. Module IFBMOD01 contains two CSECTs, IFBCST01 and IFBCST02. Here is an example of a SYSMOD containing a ++MOD statement to accomplish this:

```

++USERMOD(USR0001)      /* User modification      */.
++VER(Z038) FMID(JXY1040) /* to user application. */.
++MOD(IFBMOD01)        /* Add this module     */.
                        DISTLIB(AOSFB) /* to this DLIB at ACCEPT, */
                        LMOD(IEEFRO) /* to this LMOD at apply. */
                        CSECT(IFBCST01 /* Module has two CSECTS. */
                        IFBCST02) /* */.
...
... object deck for IFBMOD01
...

```

The following DD statement is needed at APPLY time to define the operating system load module library:

```
//LINKLIB DD DSN=SYS1.LINKLIB,DISP=OLD
```

When the SYSMOD is accepted, the following DD statement is needed to define the distribution library for this module:

```
//AOSFB DD DSN=SYS1.AOSFB,DISP=OLD
```

### Example 2: Specifying Link-Edit Utility Attributes with LEPARM

For this example, assume you have installed a product, FXY1040, packaged in RELFILE format. The package contained a module, IFBMOD01, that was identified via inline JCLIN as being installed as follows:

1. It was copied from the distribution library, AOS12, to the target system library, LPALIB.
2. It was linked with several other modules to form a load module, IFBLMDXX, in LINKLIB.
3. It was linked, by itself, to form a load module, named IFBLMDX1, in LINKLIB.

The JCLIN in the initial function was as follows:

```

//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//COPY1    EXEC PGM=IEBCOPY
//AOS12    DD DSN=SYS1.AOS12,DISP=SHR
//LPALIB   DD DSN=SYS1.LPALIB,DISP=SHR
//SYSIN    DD *
           COPY INDD=AOS12,OUTDD=LPALIB
           SELECT M=(IFBMOD01)
/*
//LINK1    EXEC PGM=IEWL,PARM='REUS'
//AOS12    DD DSN=SYS1.AOS12,DISP=SHR
//SYSLMOD  DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN   DD *
           INCLUDE AOS12(IFBMOD01)
           INCLUDE AOS12(IFBMOD0A,IFBMOD0B,IFBMOD0C)
           ENTRY  IFBMOD01
           NAME  IFBLMDXX(R)
/*
//LINK2    EXEC PGM=IEWL
//AOS12    DD DSN=SYS1.AOS12,DISP=SHR
//SYSLMOD  DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN   DD *
           INCLUDE AOS12(IFBMOD01)

```

```
NAME IFBLMDX1(R)
/*
```

The target zone now contains the following entries:

1. MOD entry for IFBMOD01, having LMOD subentries of IFBMOD01 (from the copy step), IFBLMDXX (from the first link step), and IFBLMDX1 (from the second link step).
2. LMOD entry for IFBMOD01, (created from the SELECT statement of the copy step), indicating that the load module was copied during installation. The LMOD entry does not have any link-edit utility parameters yet, because SMP/E copied IFBMOD01 from the RELFILE data sets, and, thus, had no need to obtain the link parameters.
3. LMOD entry for IFBLMDXX (created from the first link step), with link-edit attributes of REUS.
4. LMOD entry for IFBLMDX1 (created from the second link step), with link-edit attributes of STD (because no special parameters were specified).

Now assume a PTF to replace module IFBMOD01 is required. Module IFBMOD01 has link-edit utility attributes of REUS and RENT. Here is an example of a SYSMOD containing a ++MOD statement to accomplish this:

```
++PTF(UZ12345)           /* PTF                */.
++VER(Z038) FMID(FXY1040) /* for this function.    */.
++MOD(IFBMOD01)         /* Replace IFBMOD01     */.
                        DISTLIB(AOS12) /* in this DLIB.       */.
                        LEPARM(RENT /* Reentrant           */.
                        REUS) /* and reusable.        */.
...
... object deck for IFBMOD01
...
```

When the PTF is applied, SMP/E processes the LEPARM as follows:

1. Because LMOD IFBMOD01 was copied from the DLIB module, it is updated using the LEPARM values from the ++MOD MCS.
2. LMOD IFBLMDXX link-edit attributes remain as they are. This is because the link-edit utility attributes of each module within the load module have no bearing on the link-edit utility attributes of the load module; and, in this case, they are not the same. One of the other modules in IFBLMDXX must have a more restrictive set of link-edit attributes, thus forcing the load module to have that restrictive set of attributes.
3. LMOD IFBLMDX1 link-edit attributes remain as they are, for the same reason that load module IFBLMDXX attributes did not change. This is true even though load module IFBLMDX1 is composed of only the one DLIB module. As long as the load module was identified via a link-edit step, SMP/E assumes that the load module may contain multiple DLIB modules.

**Note:** The only way to change the link-edit attributes of a load module that was link-edited during initial installation is to provide JCLIN input to identify the new link-edit attributes.

## ++MOVE MCS

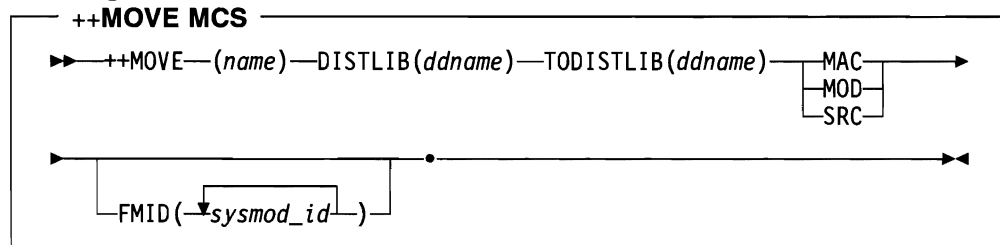
The ++MOVE MCS moves a macro, a module, a source, or a load module (and any known aliases) from one library to another. The associated target or distribution zone is automatically updated to show that the entry has been deleted from the old library and added to the new one.

### Syntax

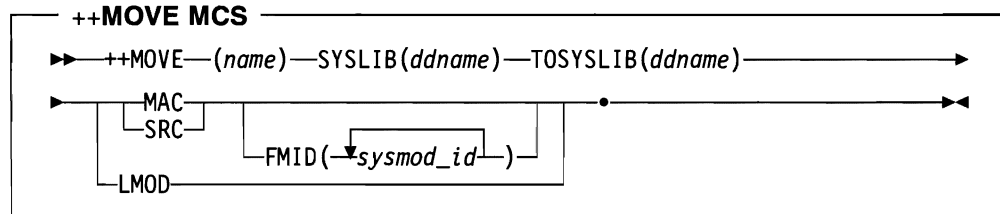
The syntax to be used depends on the type of library involved:

- Moving to a different distribution library (DISTLIB)
- Moving to a different target library (SYSLIB)

#### Moving to Another DISTLIB



#### Moving to Another SYSLIB



### Operands

#### DISTLIB

specifies the ddname of the distribution library in which the member resides.

#### FMID

specifies the FMID that owns the element. This is used when the current owner of the element is different from the FMID specified on the ++VER MCS. Up to 10 SYSMOD IDs can be specified.

**Note:** FMID is mutually exclusive with LMOD.

#### MAC, MOD, SRC, or LMOD

specifies the type of member to be moved.

- If **DISTLIB** and **TODISTLIB** are specified, **MAC**, **MOD**, or **SRC** is valid.
- If **SYSLIB** and **TOSYSLIB** are specified, **MAC**, **SRC**, or **LMOD** is valid.

**Note:** LMOD is mutually exclusive with FMID.

#### name

specifies the name of the element or load module to be moved.

**SYSLIB**

specifies the ddname of the target library in which the member resides.

**TODISTLIB**

specifies the ddname of the distribution library to which the member is to be moved. This must be specified if **DISTLIB** is specified.

**TOSYSLIB**

specifies the ddname of the target library to which the member is to be moved. This must be specified if **SYSLIB** is specified.

## Usage Notes

- The member and library operands are required. You must specify the member name, the set of libraries affected by the move, and the member type.
- ++MOVE statements must follow any ++VER and ++IF statements and must precede any element MCSs.
- Regardless of the order in which ++MOVE, ++RENAME, and ++DELETE statements are coded in a SYSMOD, they are processed in the following order:

**APPLY and ACCEPT**

1. ++MOVE
2. ++RENAME
3. ++DELETE

**RESTORE**

1. ++RENAME
2. ++MOVE

Afterwards, ++JCLIN statements are processed, and then element statements.

- You must use the FMID operand to identify all possible owners of an element to be moved.

Using the FMID operand does not imply that the owner of the element is being changed. You can change ownership only by specifying the VERSION operand on the ++VER MCS or on the element statement. Even if you are changing the owner of the element, you must specify the FMID operand, because the element is moved before its ownership is changed. For more information about the VERSION operand, see “++VER MCS” on page 591.

- A ++MOVE MCS can move a member to a given library from one library at a time. If a member exists in more than one library, you must use additional ++MOVE or ++DELETE statements to process the additional copies. For more information about ++DELETE statements, see “++DELETE MCS” on page 529.
- If a ++MOVE MCS changes the DLIB for an element that is also being replaced, the element MCS must specify the new DLIB.

**Example: Moving a Module and a Load Module**

Assume IBM ships you a PTF that moves module MODAA from its current distribution library to a new library, and also moves load module LMODA from its current target library to a new one. Here is an example of a SYSMOD containing ++MOVE statements that makes these changes:

```
++PTF(UR01234)          /* Identify the PTF number */.  
++VER(Z038) FMID(HXY1300) /* for MVS function HXY1300.*/.  
++IF (ESY1300) THEN    /* If ESY1300 is installed */  
  REQ(UR12399)         /* UR12399 is required. */.  
++MOVE (MODAA)         /* Move module MODAA */  
  DISTLIB(AOS11)      /* from DLIB AOS11 */  
  TODISTLIB(AOSXX) MOD /* to DLIB AOSXX. */.  
++MOVE (LMODA)        /* Move load module LMODA */  
  SYSLIB(LINKLIB)     /* from LINKLIB */  
  TOSYSLIB(LPALIB) LMOD /* to LPALIB. */.  
++JCLIN                /* JCLIN follows. */.  
  .  
  .  
  .  
++MOD(MODAA) DISTLIB(AOSXX) /* Element MCS statements. */.
```

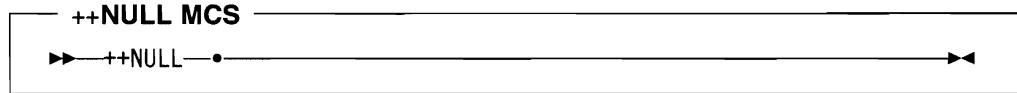


---

## ++NULL MCS

The ++NULL MCS is valid only in the SMPHOLD data set. It provides no SMP/E function other than allowing all service tapes to be built with the same format, even though a function may have no exception SYSMOD data for a particular month.

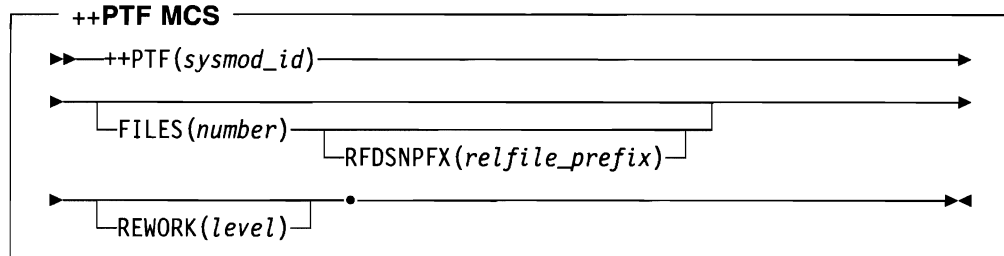
### Syntax



## ++PTF MCS

The ++PTF MCS identifies a service SYSMOD. This type of SYSMOD can replace or update elements in target and distribution libraries, such as for a permanent correction, or it can add new elements. All other MCSs for this SYSMOD follow this header MCS. For more information about packaging a PTF, see the *Standard Packaging Rules for MVS-Based Products* manual.

### Syntax



### Operands

#### FILES

specifies the number of relative files belonging to this PTF. It can be a decimal number from 1 to 9999. For information about packaging SYSMODs in relative files, see the *Standard Packaging Rules for MVS-Based Products* manual.

#### Notes:

1. Although SMP/E allows you to package PTFs in relative files, they are not generally packaged in this format.
2. If a packager uses a high-level qualifier on RELFILE data sets, the RFDSNPFX operand on the header MCS (not the RFPREFIX operand on the RECEIVE command) **must** be used to identify that high-level qualifier.

#### REWORK

specifies the level of this SYSMOD, which has been reworked for minor changes. Up to eight numeric characters can be specified.

For SYSMODs supplied by IBM, the REWORK level is *yyyyddd*, where *yyyy* is the year the SYSMOD was reworked and *ddd* is the Julian date.

REWORK allows an updated SYSMOD to be automatically received again, as long as it is more recent than the version that has already been received. This takes the place of rejecting the SYSMOD and receiving it again.

**Note:** If a SYSMOD appears more than once in the SMPPTFIN data set, the first occurrence may be received. However, none of the subsequent versions of the SYSMOD are received, even if their rework level is higher than the one for the first version of the SYSMOD. (Message GIM40001 is issued for each of the subsequent versions of the SYSMOD.)

#### RFDSNPFX

identifies to SMP/E the prefix used in the relative file data set names for this SYSMOD. SMP/E uses this prefix when allocating data set names for the SYSMOD's relative files during RECEIVE processing.

- This operand can be specified only if the FILES operand is also specified.
- The RFDSNPFX value specified on the MCS statement must match the actual prefix used in the data set names for the associated relative files.

For example, if the names of the relative files created for a SYSMOD start with "IBM," as in `IBM.sysmod_id.F1`, the header MCS statement for the SYSMOD must specify `RFDSNPFX(IBM)` so SMP/E knows which prefix to use when allocating the data set names for the SYSMOD's relative files during RECEIVE processing.

- Following standard data set naming conventions, the prefix can be from 1 to 8 alphanumeric or national (\$, #, @) characters or a dash (-).

To enable full RACF protection for tape data sets and to keep the tape header within the 17-character limit (including periods), you should limit the prefix to 1 to 3 characters. If the name exceeds the 17-character limit, only the rightmost 17 characters are written to the tape header label.

*sysmod\_id*

specifies a unique 7-character system modification identifier for the PTF. For more information, see "Naming Conventions for SYSMODs" on page 809.

## Usage Notes

If you want to update IBM-supplied code, you should use the ++USERMOD MCS rather than the ++PTF MCS. For more information, see "++USERMOD MCS" on page 589.

## Example: ++PTF to Fix One APAR

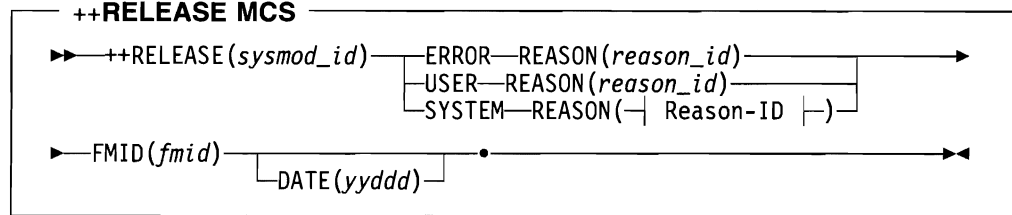
A PTF is required that replaces module IFBMOD01 for function FXY1040. The prerequisite service SYSMOD for the module is PTF UZ00004. The APAR incident fixed by this PTF is AZ12345. Here is an example of a SYSMOD containing a ++PTF statement to accomplish this:

```
++PTF(UZ00006)          /* Identify the PTF number. */.
++VER(Z038) FMID(FXY1040) /* for MVS function FXY1040 */
                    PRE(UZ00004) /* with a prerequisite. */
                    SUP(AZ12345) /* Fixes one APAR. */.
++MOD(IFBMOD01)        /* This module */
                    DISTLIB(AOSFB) /* from this DLIB. */.
...
... object deck for IFBMOD01
...
```

## ++RELEASE MCS

The ++RELEASE MCS removes a previously held SYSMOD from *exception SYSMOD* status. ++RELEASE statements are processed by the RECEIVE command.

### Syntax



SYSTEM Reason IDs Used by IBM:	
ACTION	
AO	
DELETE	
DEP	
DOC	
EC	
EXRF	
FULLGEN	
IOGEN	
MSGSKEL	
MVSCP	

### Operands

#### ERROR, SYSTEM, or USER

specifies the hold category from which the SYSMOD is to be removed. You must specify one of these categories.

**ERROR** An APAR reported an error in the SYSMOD. The SYSMOD should not be processed until the APAR is resolved. A PTF held for this reason is also called a *program error PTF*, or PE-PTF. SMP/E automatically releases the SYSMOD when a SYSMOD that supersedes the APAR is processed. Error holds can be provided in a separate HOLD file or data set, such as SMPHOLD.

**Note:** ERROR can also be specified as ERR.

**SYSTEM** Special action outside normal SMP/E processing is required for the SYSMOD. Examples are SYSMODs requiring a SYSGEN after they are installed, or SYSMODs requiring the installation of an associated engineering change (EC) level. System holds can appear in the SYSMOD itself or in a separate HOLD input file.

**Note:** SYSTEM can also be specified as SYS.

**USER** The SYSMOD requires special processing because of a decision you have made. User holds can be provided in a separate hold file or data set, such as SMPHOLD.

**DATE**

specifies the date that the ++HOLD MCS was generated.

**FMID**

specifies the FMID to which the held SYSMOD is applicable. This information allows SMP/E to receive only those statements associated with FMIDs defined in the user's global zone. This field is required.

**REASON**

identifies the HOLD reason ID that is to be removed from the SYSMOD. This field is required.

- An *error reason ID* is the number of the APAR that caused the SYSMOD to be placed in exception status.
- A *system reason ID* is a 1- to 7-character string used to identify some special processing the SYSMOD requires. These are the specific values currently used by IBM:

<b>ID</b>	<b>Explanation</b>
<b>ACTION</b>	The SYSMOD needs special handling before or during APPLY processing, ACCEPT processing, or both.
<b>AO</b>	The SYSMOD may require action to change automated operations procedures and associated data sets and user exits in products or in customer applications. The PTF cover letter describes any changes (such as to operator message text, operator command syntax, or expected actions for operator messages and commands) that can affect automation routines.
<b>DELETE</b>	The SYSMOD contains a ++DELETE MCS, which deletes a load module from the system.
<b>DEP</b>	The SYSMOD has a software dependency.
<b>DOC</b>	The SYSMOD has a documentation change that should be read before the SYSMOD is installed.
<b>EC</b>	The SYSMOD needs a related engineering change.
<b>EXRF</b>	The SYSMOD must be installed in both the active and the alternative MVS/XA* Extended Recovery Facility (XRF) systems at the same time to maintain system compatibility. (If you are not running XRF, you should bypass this reason ID.)
<b>FULLGEN</b>	The SYSMOD needs a complete system or subsystem generation to take effect.
<b>IOGEN</b>	The SYSMOD needs a system or subsystem I/O generation to take effect.
<b>MSGSKEL</b>	This SYSMOD contains message changes that must be compiled for translated versions of the message changes to become operational on extended TSO consoles.

If you want to use translated versions of the messages, you must run the message compiler once for the library containing the English message outlines, and once for each additional language you want to be available on your system. For details, see the *MVS/ESA Planning: Operations* manual.

If you want to use **only** the English version of the messages, you do not need to run the message compiler. You should bypass this reason ID.

**MVSCP** The SYSMOD requires the MVS configuration program to be run for the change to take effect.

- A *user reason ID* is defined by the user.

For additional information, see “Naming Conventions for HOLD Reason IDs and HOLD Classes” on page 807.

#### *sysmod\_id*

specifies that SMP/E is to remove the identified SYSMOD from *exception SYSMOD* status. This operand is required.

## Usage Notes

- ++RELEASE statements are not allowed within a SYSMOD. They are allowed only in SMPHOLD.
- ++RELEASE statements **unconditionally** remove a SYSMOD from exception status and should, therefore, be used with caution. To install a SYSMOD that is currently in exception status, you should probably not create and process a ++RELEASE statement, but rather use the appropriate BYPASS operand of the APPLY or ACCEPT command.
- ++RELEASE statements do not affect ++HOLD statements within a SYSMOD (internal HOLDDATA). However, SMP/E can ignore this internal HOLDDATA during APPLY or ACCEPT processing if **BYPASS(HOLDSYS)** or **BYPASS(HOLDUSER)** is specified.

## Examples

The following examples are provided to help you use the ++RELEASE MCS:

- “Example 1: Removing a SYSMOD from HOLDUSER Status”
- “Example 2: Incorrect Use of ++RELEASE” on page 579

### Example 1: Removing a SYSMOD from HOLDUSER Status

Here is an example of a ++HOLD statement that holds the PTF until after some event (such as new hardware) occurs:

```
++HOLD    (UZ12345)          /* Put this PTF          */
          FMID(FXY1040)     /* for this function     */
          USER              /* into hold user status */
          REASON(CPU0A)     /* for CPU 0A update.    */
          COMMENT(I DO NOT WANT THIS TO GO ON
                  UNTIL AFTER THE CPU CHANGE) /* */.
```

When the CPU change is made, the following sample ++RELEASE statement allows the PTF to be installed:

```
++RELEASE (UZ12345)        /* Remove this PTF      */
          FMID(FXY1040)     /* for this function     */
          USER              /* from hold user status */
          REASON(CPU0A)     /* for CPU 0A update.    */
```

That PTF is now eligible for normal installation.

**Example 2: Incorrect Use of ++RELEASE**

Assume the following ++HOLD MCS was processed as part of the normal preventive service installation:

```

++HOLD   (UZ12345)           /* Put this PTF           */
          FMID(FXY1040)      /* for this function      */
          ERROR              /* into hold error status */
          REASON(AZ12345)    /* for the APAR.         */
          COMMENT(WHEN RUNNING PRODUCT XYZ and OC4
                  ABEND MAY OCCUR) /*                          */

```

You are running one system with product XYZ installed and one without product XYZ. The PTF provides a fix for another problem you are encountering on the system without product XYZ; so you want to install this PTF on that system. Here is an example of a ++RELEASE statement that lets you apply PTF UZ12345 without having to use the BYPASS operand:

```

++RELEASE (UZ12345)         /* Remove this PTF       */
          FMID(FXY1040)     /* for this function      */
          ERROR             /* from hold error status */
          REASON(AZ12345)   /* for the APAR.         */

```

The risk with this method of processing is that SMP/E no longer has any record of PTF UZ12345 being in exception status. Therefore, the next time any modifications are installed on the system with product XYZ installed, the PTF is installed, introducing a potential OC4 problem into that system.

The correct way to install the PTF on the system without product XYZ is to use the following command:

```

SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.*/.
APPLY    S(UZ12345)         /* Process the PTF.         */
          BYPASS(HOLDERR(AZ12345)) /* Bypass known error.*/.

```

Now the PTF is installed on one system, but SMP/E still remembers that it is in hold error status and does not allow it to be installed on any other system.

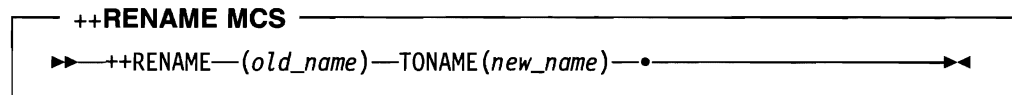
**Note:** Any other applicable operand (FORFMID, SOURCEID, and so on) can be used in place of the SELECT or S operand.

## ++RENAME MCS

During APPLY processing, the ++RENAME MCS renames a load module in all the target libraries. It also updates the LMOD subentry of the MOD entry for each module in the load module. For load modules having a CALLLIBS subentry (or having a SYSLIB allocation), SMP/E also renames the base version of the load module in the SMPLTS library.

During ACCEPT processing, if inline JCLIN was processed, the distribution zone entries are updated to reflect the ++RENAME changes. The target libraries and target zone entries are not updated.

### Syntax



### Operands

*old\_name*

specifies the name of the load module to be renamed.

**TONAME**

specifies the new name for the load module.

### Usage Notes

- There are no optional operands. You must specify the load module name and its new name.
- ++RENAME statements must follow any ++VER and ++IF statements, and must precede any element MCSs.
- Regardless of the order in which ++MOVE, ++RENAME, and ++DELETE statements are coded in a SYSMOD, they are processed in the following order:

**APPLY and ACCEPT**

1. ++MOVE
2. ++RENAME
3. ++DELETE

**RESTORE**

1. ++RENAME
2. ++MOVE

++JCLIN statements are processed next, followed by element statements.



## Example: Renaming a Single Load Module

Here is an example of a SYSMOD containing a ++RENAME statement renaming load module LMODA:

```
++PTF(UR01234)          /* Identify the PTF number. */.  
++VER(Z038) FMID(HXY1300) /* For MVS function HXY1300.*/.  
++IF (ESY1300) THEN    /* If ESY1300 is installed */  
    REQ(UR12399)        /* UR12399 is required. */.  
++RENAME (LMODA)        /* Rename load module LMODA */  
    TONAME(LMODBB)      /* to LMODBB. */.  
++JCLIN                  /* JCLIN follows. */.  
    .  
    .  
    .  
++MOD(MODAA) DISTLIB(AOSXX) /* Element MCS statements. */.
```

## ++SRC MCS

The ++SRC MCS describes a single source replacement. It must immediately precede the source definition statements when they are within the SYSMOD.

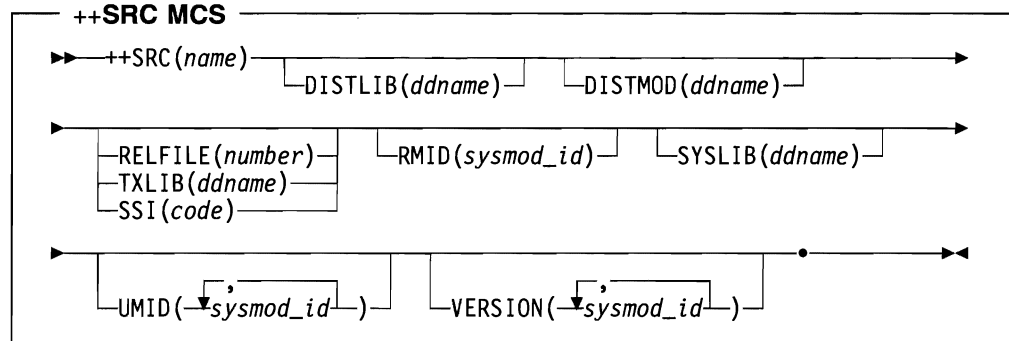
You should use the ++SRC MCS when you want to provide the source form of a module and have it get assembled when the SYSMOD is installed. If you want to provide the object form of the module, use the ++MOD MCS instead.

## Syntax

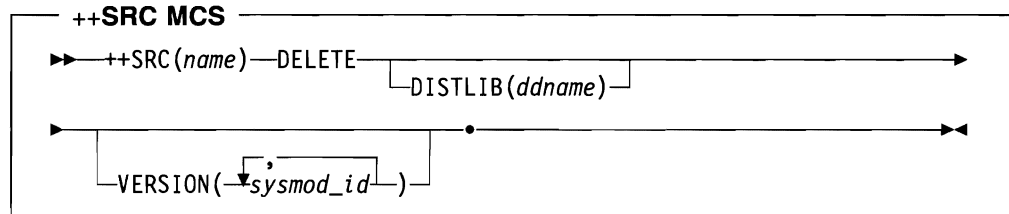
The syntax to be used depends on the processing to be done for the element:

- Adding or replacing the element
- Deleting the element

### Adding or Replacing Source



### Deleting Source



## Operands

### DELETE

indicates that the source is to be removed from the target libraries, distribution libraries, and SMP/E data sets.

**Note:** DELETE is mutually exclusive with all other operands except DISTLIB and VERSION.

### DISTLIB

specifies the ddname of the distribution library for the specified source.

**Note:** DISTLIB must be specified if the source has not been previously recorded on the target zone or distribution zone. If a SRC entry already exists in the target zone or distribution zone, and the value currently in that entry does not match that specified in the DISTLIB operand, the SYSMOD is not applied or accepted, unless that SYSMOD also used the ++MOVE MCS to change the DISTLIB to that new value.

**DISTMOD**

specifies the ddname of the link-edit distribution library for the assembled source code. During ACCEPT processing, the object code from the assembler is link-edited to the library specified.

*name*

specifies the name of the source in the distribution library and, optionally, in the target library. The name can contain any alphanumeric characters and \$, #, @, or hex C0.

**RELFILE**

identifies which relative file associated with the SYSMOD contains this element. This operand is required if you provide the element in RELFILE format, rather than inline or in a TXLIB data set.

**Notes:**

1. SMPTLIB cannot be used as a value on the TXLIB operand.
2. RELFILE is mutually exclusive with TXLIB.

**RMID**

specifies the last SYSMOD that **replaced** this source. This operand can be used only in a service-updated function, and the specified PTF must be integrated into the function.

**SSI**

specifies eight hexadecimal digits of system status information. This information is placed in the directory of the target system library or SMPMTS or SMPSTS during APPLY processing, and in the distribution library during ACCEPT processing, as four packed hexadecimal bytes of user data. See the IEBUPDTE program description in the utilities manual for your operating system.

**Note:** This operand is ignored if text is located in a library, as is the case when either the RELFILE or TXLIB operand is specified.

**SYSLIB**

specifies the ddname of the target library, if the source module exists in one. APPLY and RESTORE processing update this library.

**Note:** If a SRC entry already exists in the target zone or distribution zone and the value currently in that entry does not match that specified in the SYSLIB operand, SMP/E ignores the SYSLIB value in the SYSMOD being installed, unless that SYSMOD also used the ++MOVE MCS to change the SYSLIB to that new value.

**TXLIB**

is the ddname of the partitioned data set containing the source. This operand is required if the module is provided in a data set the users have access to, rather than inline or in RELFILE format.

**Notes:**

1. SMPTLIB cannot be used as a value on the TXLIB operand.
2. TXLIB is mutually exclusive with LKLIB and RELFILE.

**UMID**

specifies the UMIDs of the source. This operand can be used only in function SYSMODs and specifies the PTF service level of the source (the set of SYSMODs that have updated this source since it was last replaced).

**VERSION**

specifies one or more function SYSMODs that currently contain the element. The function containing the ++SRC MCS takes over ownership of the element from the specified functions.

When **VERSION** is specified on an element statement, it overrides any **VERSION** operand values that might be specified on the ++VER MCS.

**Usage Notes**

- If the source is packaged inline, it must immediately follow the ++SRC MCS and must not contain any records starting with ++. Neither **RELFILE** nor **TXLIB** can be specified.
- If the source is in a **TXLIB** data set, the ddname specified on the **TXLIB** operand is required during **APPLY** and **ACCEPT** processing.
- For information about packaging SYSMODs in **RELFILE**, **TXLIB**, or inline format, see the *Standard Packaging Rules for MVS-Based Products* manual.

**Example: Adding a New Source to the System**

A replacement for the source **IFBSRC01** is in a partitioned data set referred to by the ddname **REPLACE**. The distribution library for the source is **SYS1.IFBSRC**; **SYS1.AOS23** is the distribution library for the module, **IFBSRC01**, resulting from the assembly of the source, **IFBSRC01**.

Here is an example of a SYSMOD containing a ++SRC statement that causes SMP/E to install the source code, assemble it, and save the resulting module in the DLIBs:

```
++USERMOD(USR0001)      /* User modification      */.
++VER(Z038) FMID(FXY1040) /* for user function in MVS.*/.
++SRC(IFBSRC01)         /* Replace source      */.
                        DISTLIB(IFBSRC) /* in this DLIB.      */.
                        DISTMOD(AOS23) /* MOD goes in this DLIB. */.
                        TXLIB(REPLACE) /* Replacement SRC is here. */.
```

The following DD statements are required when the SYSMOD is applied:

```
//REPLACE DD DSN=...
//SMPSTS DD DSN=SYS1.SMPSTS,DISP=OLD
```

plus whatever other DD statements are required based on which load modules the assembled source is to be linked to. These load modules should be identified via **JCLIN**, either as a separate step or within the SYSMOD itself. Assuming the load module was composed of only this one module, the following ++JCLIN MCS can be added to the SYSMOD after the ++VER statement.

```
++JCLIN.                /* JCLIN to get SRC linked. */
//JOB1   JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1  EXEC PGM=IEBCOPY
//AOS23  DD DSN=SYS1.AOS23,DISP=SHR
//LPALIB DD DSN=SYS1.LPALIB,DISP=SHR
//SYSIN  DD *
        COPY INDD=AOS23,OUTDD=LPALIB
        SELECT M=(IFBSRC01)
/*
```

In this case, you also need the following DD statement when the SYSMOD is applied:

```
//LPALIB DD DSN=SYS1.LPALIB,DISP=OLD
```

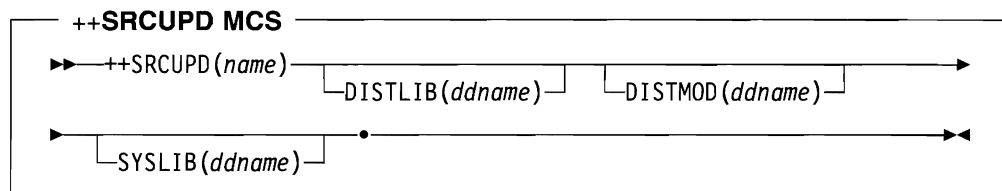
The following DD statements are required when the SYSMOD is accepted:

```
//REPLACE DD DSN=...
//IFBSRC  DD DSN=SYS1.IFBSRC,DISP=OLD
//AOS23   DD DSN=SYS1.AS023,DISP=OLD
```

## ++SRCUPD MCS

The ++SRCUPD MCS describes a single set of source update statements within a PTF, an APAR fix, or a USERMOD. It must immediately precede the source update statements within the SYSMOD.

## Syntax



## Operands

**DISTLIB**

specifies the ddname of the distribution library for the specified source.

**Note:** **DISTLIB** must be specified if the source has not been previously recorded on the target zone or distribution zone. If a SRC entry already exists in the target zone or distribution zone, and the value currently in that entry does not match that specified in the **DISTLIB** operand, the SYSMOD is not applied or accepted.

**DISTMOD**

specifies the ddname of the link-edit distribution library for the assembled source code. During **ACCEPT** processing, the object code from the assembler is link-edited to the library specified.

*name*

specifies the name of the source in the distribution library and, optionally, in the target library. The name can contain any alphanumeric characters and \$, #, @, or hex C0.

**SYSLIB**

specifies the ddname of the target library, if the source exists in one. **APPLY** and **RESTORE** processing update this library.

## Usage Notes

- If a SYSMOD containing a ++SRCUPD statement attempts to change the ownership (FMID) of the element (via the **VERSION** operand), the SYSMOD cannot be installed.
- The changes for the source must immediately follow the ++SRCUPD MCS and must not contain any records starting with ++.
- The only IEBUPDTE control statements allowed in a SYSMOD are **./ CHANGE** and **./ ENDUP**.
- The only IEBUPDTE **CHANGE** operand that SMP/E checks is **NAME**, which must specify the same element as the ++SRCUPD MCS. Other **CHANGE** operands may produce undesired results and are used at your own risk. For example, if you code **UPDATE=INPLACE**, SMP/E may update the distribution

library. Once the distribution libraries are changed, there is no way to remove the updates.

- When processing multiple updates to the same lines in a given source module, SMP/E uses the ./ CHANGE statement from the last update to the lines.
- SMP/E does not support a continuation of the ./ CHANGE statement.

## Examples

The following examples are provided to help you use the ++SRCUPD MCS:

- “Example 1: Updating an Existing Source”
- “Example 2: Making Subsequent Source Updates”

### Example 1: Updating an Existing Source

Here is an example of a SYSMOD containing a ++SRCUPD statement to make a modification to an IBM source module, in this case module JESMOD01, which is part of product EJS1102:

```

++USERMOD(MY00001)      /* User modification      */.
++VER(Z038) FMID(EJS1102) /* for MVS JES.          */.
                        PRE(UZ12345) /* Current service level. */.
++SRCUPD(JESMOD01)     /* Update this JES MOD.  */.
                        /* DISTLIB already known. */.

./ CHANGE NAME=JESMOD01
      LA  R1,MYPARM      00001000
      BALR MYPGM         00001100
MYPGM  EQU  *           00500000
      ...               00500100
      ...               00500200
      B   R14           00500300

```

### Example 2: Making Subsequent Source Updates

Assume that, after installing the modification given in the first example, you need to make another modification. Assume the following lines had to be changed:

```

      ...               00500150
      BALR MYPGM2       00500160
MYPGM2 EQU  *           00600000
      ...               00600100
      ...               00600200
      B   R14           00600300

```

You have two choices as to the method of building the second modification:

1. The first method is to build the second SYSMOD to contain only the newly changed lines of code, and then specify the PRE operand in both the current IBM service level and the previous user modification.
2. The second method is to build the second SYSMOD to contain all the user modifications to this module; then specify the PRE operand in the current IBM service level and supersede the previous user modification.

Here is an example of a SYSMOD for the first method:

```

++USERMOD(MY00002)      /* User modification      */.
++VER(Z038) FMID(EJS1102) /* for MVS JES.          */.
      PRE(UZ12345      /* Current service level  */.
      MY00001)        /* plus previous USERMOD.*/.
++SRCUPD(JESMOD01)     /* Update this JES MOD.  */.
      /* DISTLIB already known. */.

./ CHANGE NAME=JESMOD01
      ...
      BALR MYPGM2      00500150
MYPGM2 EQU *          00500160
      ...              00600000
      ...              00600100
      ...              00600200
      B R14            00600300

```

Here is an example of a SYSMOD for the second method:

```

++USERMOD(MY00002)      /* User modification      */.
++VER(Z038) FMID(EJS1102) /* for MVS JES.          */.
      PRE(UZ12345      /* Current service level. */.
      SUP(MY00001)     /* Supersede previous usermod.*/.
++SRCUPD(JESMOD01)     /* Update this JES MOD.  */.
      /* DISTLIB already known. */.

./ CHANGE NAME=JESMOD01
      LA R1,MYPARM      00001000
      BALR MYPGM        00001100
MYPGM EQU *            00500000
      ...              00500100
      BALR MYPGM2      00500160
      ...              00500200
      B R14            00500300
      ...              00500150
MYPGM2 EQU *            00600000
      ...              00600100
      ...              00600200
      B R14            00600300

```

The advantages to the second method are that all modifications to one source are contained in a single SYSMOD, and if that SYSMOD has to be restored or re-applied, processing is much more efficient.

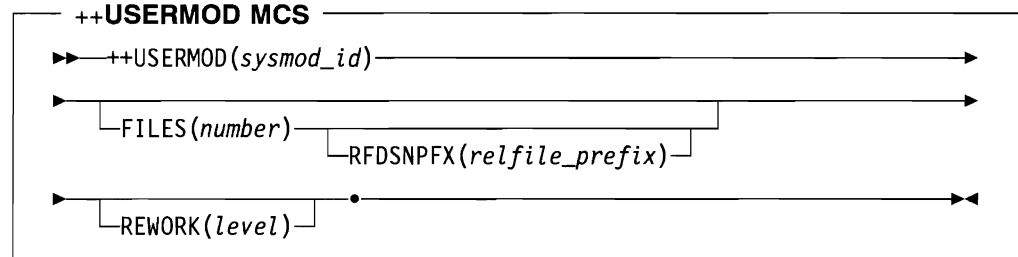
**Note:** The SMP/E ++SRCUPD MCS is the same for both methods; only the ++VER MCS and the actual update control cards change.



## ++USERMOD MCS

The ++USERMOD MCS identifies a user modification. This type of SYSMOD can be used to add user-defined functions or to replace or update elements for IBM-supplied code in the target or distribution libraries. All other MCSs for this SYSMOD follow this header MCS. For more information about packaging a USERMOD, see the *SMP/E R8.1 User's Guide*.

### Syntax



### Operands

#### FILES

specifies the number of relative files belonging to this USERMOD. It can be a decimal number from 1 to 9999. For information about packaging SYSMODs in relative files, see the *Standard Packaging Rules for MVS-Based Products* manual.

#### Notes:

1. Although SMP/E allows you to package USERMODs in relative files, they are not generally packaged in this format.
2. If a packager uses a high-level qualifier on RELFILE data sets, the RFDSNPFX operand on the header MCS (not the RFPREFIX operand on the RECEIVE command) **must** be used to identify that high-level qualifier.

#### REWORK

specifies the level of this SYSMOD, which was reworked for minor changes. Up to eight numeric characters can be specified.

For SYSMODs supplied by IBM, the REWORK level is *yyyyddd*, where *yyyy* is the year the SYSMOD was reworked and *ddd* is the Julian date.

REWORK allows an updated SYSMOD to be automatically received again, as long as it is more recent than the version that has already been received. This takes the place of rejecting the SYSMOD and receiving it again.

**Note:** If a SYSMOD appears more than once in the SMPPTFIN data set, the first occurrence may be received. However, none of the subsequent versions of the SYSMOD are received, even if their rework level is higher than the one for the first version of the SYSMOD. (Message GIM40001 is issued for each of the subsequent versions of the SYSMOD.)

**RFDSNPFX**

identifies to SMP/E the prefix used in the relative file data set names for this SYSMOD. SMP/E uses this prefix when allocating data set names for the SYSMOD's relative files during RECEIVE processing.

- This operand can be specified only if the FILES operand is also specified.
- The RFDSNPFX value specified on the MCS statement must match the actual prefix used in the data set names for the associated relative files.

For example, if the names of the relative files created for a SYSMOD start with "IBM," as in `IBM.sysmod_id.F1`, the header MCS statement for the SYSMOD must specify `RFDSNPFX(IBM)` so SMP/E knows which prefix to use when allocating the data set names for the SYSMOD's relative files during RECEIVE processing.

- Following standard data set naming conventions, the prefix can be from 1 to 8 alphanumeric or national (\$, #, @) characters or a dash (-).

To enable full RACF protection for tape data sets and to keep the tape header within the 17-character limit (including periods), you should limit the prefix to 1 to 3 characters. If the name exceeds the 17-character limit, only the rightmost 17 characters are written to the tape header label.

*sysmod\_id*

specifies a unique 7-character system modification identifier for the USERMOD. For more information, see "Naming Conventions for SYSMODs" on page 809.

**Usage Notes**

If you have updated an element, SYSMODs other than functions cannot replace it unless you explicitly allow them to by bypassing MODID checking. However, if you install a function, it may overlay your user modifications. SMP/E issues a warning message when it detects this condition.

**Example: ++USERMOD to Update Source**

A source (IFBSRC02) owned by function SYSMOD FXY1040 is to be modified. Your modification requires a service level provided UZ00007; you are only updating, not replacing, the source. You have chosen a SYSMOD ID of MY00001. Here is an example of a SYSMOD containing a ++SRCUPD statement that accomplishes this:

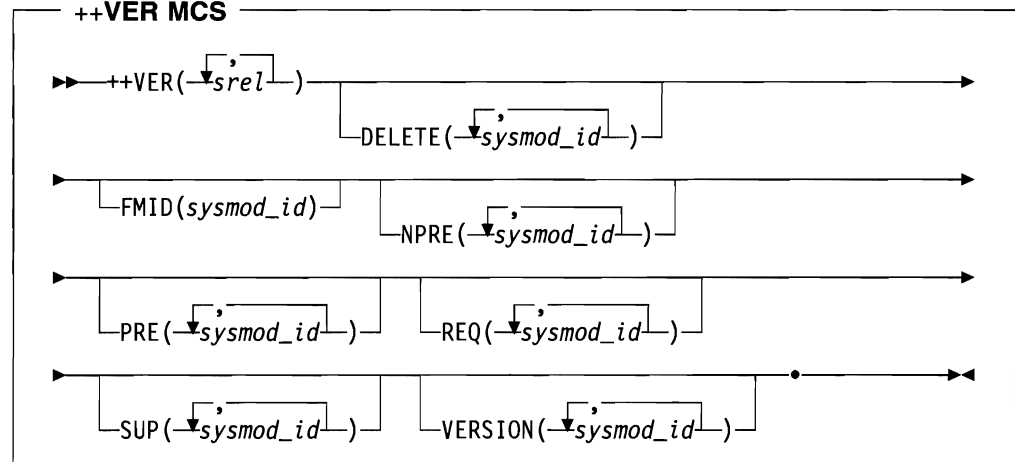
```
++USERMOD(MY00001)      /* USERMOD number */.
++VER(Z038) FMID(FXY1040) /* for function FXY1040 */
                       PRE(UZ00007) /* at this service level. */.
++SRCUPD(IFBSRC02)     /* Update this source */
                       DISTLIB(IFBSRC) /* in this DLIB. */.
./ CHANGE NAME=IFBSRC02
...
... update control cards
...
```

For additional examples of USERMODs, see "++SRCUPD MCS" on page 586 and the *SMP/E R8.1 User's Guide*.

## ++VER MCS

The ++VER MCS describes the environment required for receiving and installing a SYSMOD. A SYSMOD must contain a separate ++VER MCS for each environment to which it applies. At least one ++VER MCS must be present in a SYSMOD, and a maximum of 255 ++VER statements are allowed for each SYSMOD.

### Syntax



### Operands

#### DELETE

indicates which function SYSMODs should be deleted when this function is installed. These functions are permanently deleted and cannot be restored.

**DELETE** can be specified only in function SYSMODs.

The same SYSMOD can be specified on both **DELETE** and **SUP**. This cleans up entries for the deleted function, and, at the same time, allows SYSMODs that name the deleted function as a requisite to still be installed.

SYSMODs specified in the **DELETE** operand do not have to be specified in **VERSION** operands of ++VER, ++MAC, ++SRC, or ++MOD statements.

For additional information on the effect of **DELETE** during **APPLY** and **ACCEPT** processing, see "Deleted SYSMODs" on page 83 and "Deleted SYSMODs" on page 35.

#### FMID

FMID identifies the function to which a SYSMOD applies. FMID must be specified for all SYSMODs except base functions.

The following considerations relate to the FMID operand:

- Unlike prerequisites specified by the **PRE** operand, the functional prerequisite specified by the **FMID** operand is satisfied only by the specified SYSMOD. It is not satisfied by another SYSMOD that supersedes that function.
- When specified on the ++VER MCS for a function, FMID defines the function as a dependent function. In this case, FMID indicates that the ele-

ments supplied by the dependent function SYSMOD are functionally higher than the specified base function.

A function cannot be both a base function and a dependent function. Therefore, if a base function contains more than one ++VER MCS, none of them can specify the FMID operand. Likewise, if a dependent function contains more than one ++VER MCS, all of them must specify the FMID operand.

- When specified on the ++VER MCS for a non-function SYSMOD, FMID indicates the functional level of all elements in the SYSMOD.
- SMP/E RECEIVE processing does not receive a dependent function unless the FMID of the base is already present in the global zone or BYPASS(FMID) is specified. For more information on packaging of dependent SYSMODs, refer to RECEIVE processing in Chapter 12, “The RECEIVE Command.”

**NPRE**

indicates which function SYSMODs cannot exist in the same zone as this function. These are negative prerequisite SYSMODs. The current SYSMOD cannot be applied or accepted if any of the listed SYSMODs are already present.

This operand has no effect on RECEIVE eligibility.

**NPRE** can only be specified within a function SYSMOD.

**PRE**

indicates which SYSMODs are prerequisites for this SYSMOD. A prerequisite SYSMOD must either be already installed, or must be installed concurrently with this SYSMOD. For additional information on how prerequisites are resolved during APPLY and ACCEPT processing, see “Processing” on page 77 and “Processing” on page 28.

If a SYSMOD replaces an existing element, the PRE operand must specify the previous SYSMOD that replaced the element (RMID) and all the SYSMODs that have updated the element (UMIDs) since it was last replaced.

If a SYSMOD updates an existing element, the PRE operand must specify the previous SYSMOD that replaced the element. It should also specify the last SYSMOD that updated the element since then.

This operand has no effect on RECEIVE eligibility.

**REQ**

indicates which SYSMODs are requisites for this SYSMOD. The specified SYSMOD must either be already installed, or must be installed concurrently with this SYSMOD. If the specified SYSMOD also specifies this SYSMOD as a requisite, these two SYSMODs are corequisites, and neither can be installed independently; they must be installed within the same APPLY and ACCEPT command. For additional information on how requisites are resolved during APPLY and ACCEPT processing, see “Processing” on page 77 and “Processing” on page 28.

This operand has no effect on RECEIVE eligibility.

**srel**

specifies the system or subsystem release on which the SYSMOD can be installed. The SREL must contain 4 characters, usually 1 alphabetic character followed by 3 numeric characters. These are the systems and subsystems defined by IBM, with their SRELs:

<b>System</b>	<b>SREL</b>
MVS	Z038
CICS	C150
NCP	P004
IMS, DB2	P115

The SREL is used during RECEIVE processing to determine whether a SYSMOD should be received. For more information on how the SREL operand is processed, see "Processing" on page 249.

**SUP**

indicates which SYSMODs are superseded (contained in and replaced) by this SYSMOD. For example, it may specify one or more APARs fixed in the element modifications supplied with this SYSMOD.

For functions, the same SYSMOD can be specified on both DELETE and SUP. This cleans up entries for the deleted function, and, at the same time, allows SYSMODs that name the deleted function as a requisite to still be installed.

**VERSION**

indicates functions whose elements should be considered functionally lower than the elements contained in this SYSMOD. It specifies one or more function SYSMODs that currently contain the element. The function containing the ++VER MCS takes over ownership of all the elements from the specified functions.

When **VERSION** is specified on an element statement, it overrides any VERSION operand values specified on the ++VER MCS.

**Note:** A SYSMOD containing an element update (++MACUPD, ++SRCUPD, or ++ZAP) cannot change the ownership of the element. The ownership can be changed (via the ++VER VERSION operand) only if the SYSMOD provides a replacement for the element.

**Usage Notes**

- You can build a SYSMOD that can be processed by previous versions of SMP, as well as this version of SMP/E. For service SYSMODs, this construction requires at least two ++VER statements, one processable by previous versions of SMP and the other processable by this version of SMP/E. The SRELs in these ++VER statements must be different so the SYSMOD can be processed correctly by the applicable version of SMP or SMP/E.
- A SYSMOD cannot contain multiple ++VER statements that have identical SREL and FMID values, because SMP/E would not be able to determine which ++VER MCS to use in doing the remaining applicability checking during APPLY and ACCEPT processing.
- You cannot specify the same SYSMOD more than once on a single ++VER operand. Likewise, you generally cannot specify the same SYSMOD on more than one operand. However, you can specify the same SYSMOD on

VERSION and another operand (except FMID). You can also specify the same SYSMOD on the DELETE and SUP operands.

- Corequisite SYSMODs (which are related through the REQ operand) that are applicable to the same FMID **cannot** have elements in common. Because the REQ operand implies no service hierarchy, SMP/E cannot determine which SYSMOD has the highest service level of the common elements. When the relationship between the SYSMODs containing the common elements is defined through the REQ operand, SMP/E issues an error or warning message.
  - If the requisites have an element in common and each contains a replacement for the element, ID check processing fails and neither of the requisites is installed, unless **BYPASS(ID)** is specified.
  - If the requisites have an element in common and each contains an update for the element, ID check processing issues a warning message for the requisites. Generally, the requisites are both installed. However, SMP/E does not allow multiple ZAPs for the same module to be processed by the same APPLY command. If this is the case, neither of the requisites is installed; they must be processed by separate APPLY commands.

## Examples

The following examples are provided to help you use the ++VER MCS:

- “Example 1: Defining Base and Dependent Functions”
- “Example 2: Defining Intersecting Dependent Functions” on page 595
- “Example 3: Deleting a Previous Level of a Function” on page 595
- “Example 4: Deleting a Function without Replacing It (Dummy Delete)” on page 596

### Example 1: Defining Base and Dependent Functions

Assume you want to package one of your user applications as a function so you can use SMP/E in installing and maintaining it. You also have an optional enhancement to that product you want to package. The base function has two modules, USRMOD01 and USRMOD02, which are in link-edit format and reside in the library pointed to by the USRLIBXX DD statement. The optional enhancement changes USRMOD02 and depends on the base application being present. Here are examples of SYSMODs for these functions:

```

++FUNCTION(EUSR001)      /* Base application      */.
++VER(Z038)              /* for MVS version.     */.
++MOD(USRMOD01)         /* Has this MOD         */
    DISTLIB(AOS12)      /* in this DLIB.       */
    LKLIB(USRLIBXX)     /* Replacement is here */.
++MOD(USRMOD02)         /* Has this MOD         */
    DISTLIB(AOS12)      /* in this DLIB.       */
    LKLIB(USRLIBXX)     /* Replacement is here. */.

```

```

++FUNCTION(FUSR011)      /* Dependent function */.
++VER(Z038)              /* for MVS version. */.
      FMID(EUSR001)      /* Base application must be
                          present. */.
++MOD(USRM002)          /* Has this MOD */
      DISTLIB(AOS12)    /* in this DLIB. */
      LKLIB(USRLIBXX)  /* Replacement is here. */.
++MOD(USRM003)          /* Has this MOD */
      DISTLIB(AOS12)    /* in this DLIB. */
      LKLIB(USRLIBXX)  /* Replacement is here. */.

```

The dependent function, FUSR011, specifies the base function on its FMID operand. This means the base function must be present in order for the dependent function to be installed. The FMID operand also indicates that if the two functions have any elements in common, the version in the dependent product is the one that is to be installed. The dependent product has now assumed ownership of those elements.

### Example 2: Defining Intersecting Dependent Functions

Assume you want to add another dependent function for the base function defined in the preceding example. It works whether or not the other dependent function is installed. Here is an example of a SYSMOD containing a ++VER statement to define the relationship between the dependent functions:

```

++FUNCTION(FUSR012)      /* Dependent function */.
++VER(Z038)              /* for MVS version. */.
      FMID(EUSR001)      /* Base application must be
                          present. */.
      VERSION(FUSR011)  /* If present is at higher
                          functional level. */.
++MOD(USRM002)          /* Has this MOD */
      DISTLIB(AOS12)    /* in this DLIB. */
      LKLIB(USRLIBXX)  /* Replacement is here. */.
++MOD(USRM004)          /* Has this MOD */
      DISTLIB(AOS12)    /* in this DLIB. */
      LKLIB(USRLIBXX)  /* Replacement is here. */.

```

After this function is installed, module USRM002 is the version from function FUSR012, no matter what the former functional level was.

### Example 3: Deleting a Previous Level of a Function

Assume you want to provide a new level of the base function defined in the first example. It includes both of the dependent functions for the original base function. Here is an example of a SYSMOD containing a ++VER statement deleting the previous level of the function:

```

++FUNCTION(EUSR002)      /* New base application */.
++VER(Z038)              /* for MVS version. */.
      DELETE(EUSR001) /* Delete prev level. */.
++MOD(USRM001)          /* Has this MOD */.
      DISTLIB(AOS12) /* in this DLIB. */.
      LKLIB(USRLIBXX) /* Replacement is here. */.
++MOD(USRM002)          /* Has this MOD */.
      DISTLIB(AOS12) /* in this DLIB. */.
      LKLIB(USRLIBXX) /* Replacement is here. */.
++MOD(USRM003)          /* Has this MOD */.
      DISTLIB(AOS12) /* in this DLIB. */.
      LKLIB(USRLIBXX) /* Replacement is here. */.
++MOD(USRM004)          /* Has this MOD */.
      DISTLIB(AOS12) /* in this DLIB. */.
      LKLIB(USRLIBXX) /* Replacement is here. */.

```

After the new function is installed, all references to SYSMODs EUSR001, FUSR011, and FUSR012 are deleted from the target zone and distribution zone (except the SYSMOD entry for EUSR001, which indicates it was deleted by EUSR002).

This SYSMOD does not require the previous level of the base function to be installed. The DELETE operand just says that if that previous function was installed, SMP/E should delete it before installing the new level. For more information on delete processing for the APPLY and ACCEPT commands, see "Processing" on page 77 and "Processing" on page 28.

#### Example 4: Deleting a Function without Replacing It (Dummy Delete)

Assume you no longer need a particular function, and you want to delete it from your system. First, you must make sure that no other functions depend on the function you want to delete. Once you have done this, you need to define a dummy function SYSMOD that deletes the function you want to delete. You then receive, apply, and accept the dummy function, and run UCLIN to delete the SYSMOD entries for the deleted function and for the dummy function.

For example, assume you are ready to delete function MYFUNC1 using dummy function DELFUNC. MYFUNC1 is applicable to SREL Z038 and is installed in target zone TGT1 and distribution zone DLIB1. Here is an example of the dummy function:

```

++FUNCTION(DELFUNC)      /* Any valid unique SYSMOD ID. */.
++VER(Z038)              /* For SREL Z038 (MVS products). */.
      DELETE(MYFUNC1) /* Deletes MYFUNC1. */.

```

These are the commands you use to receive and install the dummy function, and to delete the SYSMOD entries for the dummy function and the deleted function:



```

SET      BDY(GLOBAL)      /* Set to global zone.      */.
RECEIVE S(DELFUNC)      /* Receive the function.    */.
SET      BDY(TGT1)       /* Set to applicable target.*/.
APPLY    S(DELFUNC)      /* Apply to delete old     */.
                               /* function.                */.
SET      BDY(DLIB1)      /* Set to applicable DLIB.  */.
ACCEPT   S(DELFUNC)      /* Accept to delete old    */.
                               /* function.                */.
SET      BDY(TGT1)       /* Set to applicable target.*/.
UCLIN.
DEL      SYSMOD(DELFUNC) /* Delete SYSMOD entries for */.
DEL      SYSMOD(MYFUNC1) /* dummy and old function.  */.
ENDUCL.
SET      BDY(DLIB1)      /* Set to applicable DLIB.  */.
UCLIN.
DEL      SYSMOD(DELFUNC) /* Delete SYSMOD entries for */.
DEL      SYSMOD(MYFUNC1) /* dummy and old function.  */.
ENDUCL      /*                          */.

```

When you accept the dummy function, SMP/E automatically deletes the associated SYSMOD entry from the global zone and the MCS entry from the SMPPTS.

To complete the cleanup, you should also use the REJECT command to delete any SYSMODs and HOLDDATA applicable to the dummy function and the old function. In addition, you should delete the FMIDs from the GLOBALZONE entry to prevent SMP/E from receiving any SYSMODs or HOLDDATA applicable to either of those functions. Here are examples of the commands you can use to do this.

```

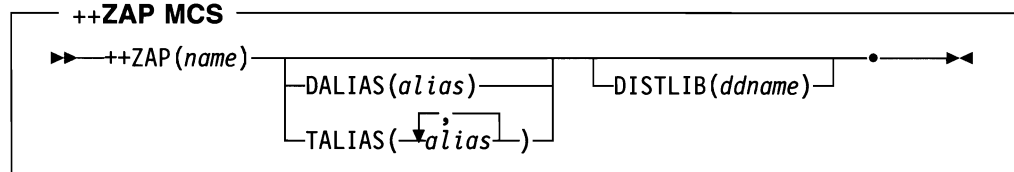
SET      BDY(GLOBAL)      /* Set to global zone.      */.
REJECT   HOLDDATA NOFMID /* Reject SYSMODs, HOLDDATA */
          DELETEFMID     /* for the deleted functions.*/
          (DELFUNC MYFUNC1) /* Delete the FMIDs from the */
                               /* GLOBALZONE entry.        */.

```

## ++ZAP MCS

The ++ZAP MCS describes a module update within a PTF, APAR fix, or USERMOD. It must precede the IMASPZAP statements within the SYSMOD.

## Syntax



## Operands

**DALIAS**

is the alias name of a module that has an alias in the distribution library, but not in the target library. This might be used if the module is included under its alias name during system generation.

**Note:** DALIAS is mutually exclusive with TALIAS.

**DISTLIB**

specifies the ddname of the distribution library for the specified module.

**Note:** This operand must be specified if the module has not been previously recorded on the target zone or distribution zone. If a MOD entry already exists in the target zone or distribution zone and the value currently in that entry does not match that specified in the DISTLIB operand, the SYSMOD is not applied or accepted.

*name*

specifies the name of the module member in the distribution library and, optionally, in the target system library. The name can contain any alphanumeric characters and \$, #, @, or hex C0.

**Note:** If the module to be updated has been assembled, specify the name of the assembled module, not the CSECT name.

**TALIAS**

identifies all the alias names of a module that has aliases in both the target and distribution libraries.

You can use TALIAS for a module that was copied from a distribution library into a target library (defined by JCLIN data as a copied module), but not for one that is link-edited (defined by JCLIN data as a link-edited module). TALIAS must be specified on the ++MOD MCS even if ALIAS was specified on the COPY SELECT statement.

**Note:** TALIAS is mutually exclusive with DALIAS.

## Usage Notes

- If a SYSMOD containing a ++ZAP statement attempts to change the ownership (FMID) of the element (via the VERSION operand), the SYSMOD cannot be installed.
- The changes for the module must immediately follow the ++ZAP MCS and must not contain any records that start with ++.
- The only IMASPZAP control statements allowed in a SYSMOD are:

ABSDUMP	DUMP	NAME	VER
ABSDUMPT	DUMPT	REP	VERIFY
BASE	IDRDATA	SETSSI	*(comment)

- An EXPAND control statement in link-edit utility format can be placed within IMASPZAP input to allow lengthening of control sections. The EXPAND statement must follow the NAME statement. For the syntax and description of the EXPAND statement, see the link-edit utility and loader manual for your operating system.
- Expand type IMASPZAP processing cannot be performed against a noneditable (NE) module.
- Any SETSSI statements placed in the input stream for expand-type IMASPZAP processing must be in a form acceptable to both IMASPZAP and the link-edit utility; that is, they must begin in column 2 or after. The SSI statements must follow the EXPAND statements.
- The name specified on the ++ZAP MCS must be the same as the name of the distribution library module. The CSECT value on the IMASPZAP NAME statement must be the same as the load module's CSECT name. That CSECT name is usually the same as the distribution library name, but it can be different. For example, if the module to be updated has been assembled, the ++ZAP statement should specify the name of the assembled module, not the CSECT name.

The LIST LMOD statement produces a target zone listing of link-edit utility control statements that might have changed the CSECT name of the member. A link-edit map may be helpful in other cases where the names differ.

- The IMASPZAP NAME statement can optionally be coded as follows:

**NAME** *csect-name*

or

**NAME** *lmod-name csect-name*

The coding of one operand assumes that operand to be a CSECT name for the module referred to in the ++ZAP statement. In this case, all load modules containing the module named in the ++ZAP statement are processed by IMASPZAP.

Two operands can be specified, in which case the second operand is assumed to be a CSECT name, as specified above. The first operand is assumed to be a valid load module containing the module named in the ++ZAP statement. In this case, only the indicated load module is processed by IMASPZAP.

- When using IMASPZAP on an assembled module, be careful: The modification identifier is updated, but not the modification of any associated macros.

It is not recommended that you use IMASPZAP to modify assembled modules. An assembled module modified by IMASPZAP does not cause updating of the distribution library during accept processing; therefore, a subsequently generated system does not contain the IMASPZAP modification.

A more satisfactory method of updating assembled modules is to update the macros that generate them.

- SMP/E processing does not save a backup copy of the nucleus during apply processing when the nucleus is modified by a SYSMOD containing a non-expand-type IMASPZAP modification.
- Only one ZAP can be applied to a module by a single APPLY command. If you need to install several ZAPs for a given module, each one must be packaged separately and installed by a separate APPLY command.
- If you need to install a ++ZAP change on a live system library and that library uses LLA and/or VLF, you will need to do some setup work before installing the change. Otherwise, the change might not take effect even after installation. See the *Initialization and Tuning Guide* appropriate to your system for guidance on:
  - Removing libraries from LLA and VLF control
  - Refreshing LLA and VLF

**Note:** Installing a change on a live system is **not** recommended.

## Examples

The examples in this section are based on the load module structure shown in Figure 85.

Load Module Name Module Name CSECT Name	Load Module Name Module Name CSECT Name
LMODA	LMODB
MOD1	MOD1
CSECT1	CSECT1
CSECT2	CSECT2
CSECT3	CSECT3
MOD2	MOD2
MOD2	MOD2
MOD3	
MOD3	
MOD4	
MOD4	

Figure 85. Load Module Structure for ++ZAP Examples

The following examples show how ++ZAP statements can be used to update modules, load modules, and CSECTs within modules:

- “Example 1: Changing All Load Modules That Contain the Same Module” on page 601

- “Example 2: Changing the Only Load Module That Contains a Given Module” on page 601
- “Example 3: Changing One of Several Load Modules That Contain a Given Module”
- “Example 4: Expanding a Module”

### Example 1: Changing All Load Modules That Contain the Same Module

Assume you want to change CSECT2 in module MOD1, which is in both LMODA and LMOB. Here is an example of a SYSMOD that accomplishes this by specifying the CSECT name on the NAME statement without including any load module names:

```
++USERMOD(MYMOD01) .
++VER(Z038) FMID(FXY1000) .
++ZAP(MOD1) .
  NAME CSECT2
  VER 000D FF4160
  REP 000D FE4160
```

### Example 2: Changing the Only Load Module That Contains a Given Module

Assume you want to change CSECT MOD3 in module MOD3, which is only in LMODA. Here is an example of a SYSMOD that accomplishes this by specifying the CSECT name on the NAME statement without including any load module names:

```
++USERMOD(MYMOD02) .
++VER(Z038) FMID(FXY1000) .
++ZAP(MOD3) .
  NAME MOD3
  VER 000A 00
  REP 000A FF
```

### Example 3: Changing One of Several Load Modules That Contain a Given Module

Assume you want to change CSECT2 in module MOD1, which is in both LMODA and LMOB. You want to update only the version in LMOB. Here is an example of a SYSMOD that accomplishes this by specifying both the load module name and the CSECT name on the NAME statement.

```
++USERMOD(MYMOD03) .
++VER(Z038) FMID(FXY1000) .
++ZAP(MOD1) .
  NAME LMOB CSECT2
  VER 0000 00
  REP 0000 FF
```

### Example 4: Expanding a Module

Assume you want to update CSECT3 with an EXPAND request. CSECT3 is in module MOD1, which is in both LMODA and LMOB. Here is an example of a SYSMOD that accomplishes this by specifying the CSECT name on the NAME statement and on an EXPAND statement.

```
++USERMOD(MYMOD04).  
++VER(Z038) FMID(FXY1000).  
++ZAP(MOD1).  
  NAME CSECT3  
  VER 000D FF  
  REP 000D FE  
  EXPAND CSECT3(4)
```

---

## Chapter 33. SMP/E PARMLIB Member Control Statements

During JCLIN processing, SMP/E scans the SMPJCLIN input, which consists of various job steps calling system utilities. To determine the structure of the target system, SMP/E specifically looks at copy steps, link-edit steps, and assembly steps.

As SMP/E scans inline assembly steps, it looks at each assembler instruction to determine whether the instruction is a macro invocation or an OPCODE. If SMP/E determines that this is a macro invocation, it builds a MAC entry in the target zone and defines the connection between the macro and the assembly in which the macro was found. As a result, when that macro is later changed by the installation of a SYSMOD, SMP/E can cause all the assemblies that used that macro to be redone.

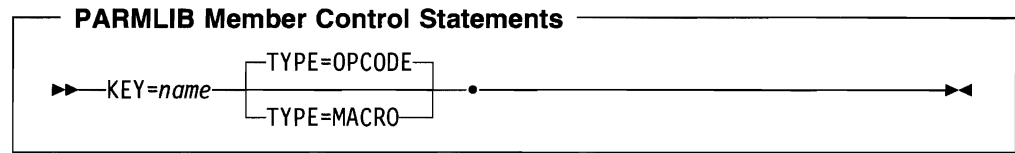
SMP/E uses PARMLIB members to determine whether an assembler instruction is a macro invocation or an OPCODE.

- Member GIMOPCDE is supplied with SMP/E and identifies all the known standard assembler OPCODEs. SMP/E assumes that any instruction not found in that list is a macro.
- You can define your own members to identify additional macro names or OPCODE values, or to change the way SMP/E should treat one or more of the character strings identified in GIMOPCDE.

You can specify the name of one of these user-defined PARMLIB members on the JCLIN command or on the ++JCLIN statement to have SMP/E pick up this additional information. The data from this other PARMLIB member is merged with the data from GIMOPCDE. If duplicate data is specified, the user-specified member has priority.

For additional information about how SMP/E processes assembler input as JCLIN, see "Processing Assembler Steps" on page 183 and "++JCLIN MCS" on page 548.

### Syntax



### Operands

#### KEY

identifies a character string for SMP/E to check for when scanning assembler entries during JCLIN processing, either during the JCLIN command or when applying a SYSMOD with inline JCLIN.

*name* can be any alphanumeric character string from one to eight characters in length.

#### TYPE

specifies how SMP/E treats that character string when it is encountered. The following values may be specified:

##### OPCODE

specifies that SMP/E treats the specified character string as a valid assembler OPCODE.

**Note:** OPCODE can also be specified as OP.

##### MACRO

indicates that SMP/E treats the specified character string as the name of a macro.

**Note:** MACRO can also be specified as MAC.

#### Notes:

1. The TYPE operand can be specified only once per control statement.
2. If the TYPE operand is not specified, the default is TYPE=OPCODE.

---

### Usage Notes

- Either a blank or a comma can be used to separate the operands.
- Comments are permitted, subject to the next restriction.
- A PARMLIB control statement **cannot** span multiple lines. The complete control statement, including comments, must be contained on a single line.
- If, during JCLIN processing, SMP/E encounters a string specified in one of the PARMLIB control statements, and the PARMLIB control statement specifies that the string is to be treated as a macro, SMP/E builds a MAC entry in the target zone and connects that MAC entry to the ASSEM entry built for the current assembly. For a further description of JCLIN processing as related to the PARMLIB members, see Chapter 8.



---

## Examples

The following examples are provided to help you use PARMLIB statements:

- “Example 1: Defining a New OPCODE for Special Assemblers”
- “Example 2: Overriding an SMP/E-Defined OPCODE”

### Example 1: Defining a New OPCODE for Special Assemblers

Some licensed programs use a version of the assembler supplied with their product rather than the standard assembler. Often, this assembler recognizes special OPCODEs. When SMP/E is processing JCLIN containing assembler steps that use these OPCODEs, you do not want SMP/E creating target zone macro entries for them. To avoid this situation, create a special PARMLIB member containing appropriate control statements to define those OPCODEs to SMP/E.

The following is an example of a PARMLIB member containing three control statements to define the three-character strings, *OP1*, *OPCD1*, and *OPCODE01* as special assembler OPCODEs:

```
KEY=OP1 ,      TYPE=OPCODE.  
KEY=OPCD1     TYPE=OP.  
KEY=OPCODE01  TYPE=OPCODE.
```

### Example 2: Overriding an SMP/E-Defined OPCODE

GIMOPCODE, the PARMLIB member supplied with SMP/E, contains a control statement identifying *PUNCH* as a character string that should be treated as an assembler OPCODE. You may have constructed a *PUNCH* macro and now want SMP/E to recognize *PUNCH* as a macro name. This can be done by adding the following control statement to a PARMLIB member and then specifying that member on the JCLIN command or ++JCLIN statement:

```
KEY=PUNCH     TYPE=MACRO /* override type to macro */.
```



---

## Chapter 34. SMP/E Data Sets

This chapter describes the data sets that are used to process SMP/E commands. These data sets are listed by the ddnames for the data sets required by each SMP/E command. According on the type of data set, you can define it with a DDDEF entry, a DD statement, or module GIMMPDFT. For more information, see “DDDEF Entry (Distribution, Target, and Global Zone)” on page 654, and Appendix C.

---

### Distribution Library (DLIB)

<b>ddname</b>	The ddname for a distribution library should match the low-level qualifier of the data set name. For example, the ddname for SYS1.AMACLIB should be AMACLIB.
<b>Use</b>	Distribution libraries (DLIBs) contain updated versions of macros, source, and modules that were shipped by IBM and stored during ACCEPT processing. They are used during system generation to build the target libraries, so you should keep them at a tested functional and service level. They are also used during RESTORE processing to replace elements in the target libraries. You must provide a DDDEF entry or DD statement for each distribution library that is being processed.
<b>Attributes</b>	Partitioned.
<b>Device</b>	Direct access only.

---

### INFILE Data Set

<b>ddname</b>	The ddname for an INFILE data set cannot be the same as the ddname for any other data sets used by SMP/E. Other than this, there are no restrictions.
<b>Use</b>	<p>The INFILE data set is a sequential data set containing a zone that is to be loaded by the ZONEIMPORT command. The ddname of this sequential data set is specified on the INFILE operand of the ZONEIMPORT command. The zone in the INFILE data set can be used in two ways:</p> <ul style="list-style-type: none"><li>• To recreate a zone that was destroyed</li><li>• To recreate a zone on another CSI data set</li></ul> <p>You must use the ZONEEXPORT command to create the INFILE data set. (This same data set is called the OUTFILE data set when it is created by the ZONEEXPORT command.)</p>
<b>Attributes</b>	<p>Sequential; no DCB parameters are required.</p> <p>If the INFILE data set is on tape, there may be more than one volume for an exported data set. Remember to specify all the volume serial numbers on the INFILE DD statement.</p>
<b>Device</b>	Tape or direct access.

### Link Library (LKLIB)

<b>ddname</b>	The ddname for a link library must match the LKLIB value on the element MCS. For example, the ddname for the link library on statement ++MOD(MODA) LKLIB(LIBX) must be LIBX.
<b>Use</b>	Link libraries contain replacements for object modules in link-edited format. They are used when the modules are provided in partitioned data sets rather than inline or in relative files.
<b>Attributes</b>	Partitioned.
<b>Device</b>	Direct access only.

---

### OUTFILE Data Set

<b>ddname</b>	The ddname for an OUTFILE data set cannot be the same as the ddname for any other data sets used by SMP/E. Other than this, there are no restrictions.
<b>Use</b>	<p>The OUTFILE data set is a sequential data set containing a zone copied by the ZONEEXPORT command. The ddname of this sequential data set is specified on the OUTFILE operand. The zone in the OUTFILE data set can be used in two ways:</p> <ul style="list-style-type: none"><li>• As a backup copy, to recreate a zone that was destroyed</li><li>• As a transportable copy, to recreate a zone on another CSI data set</li></ul> <p>You must use the ZONEIMPORT command to process the OUTFILE data set. (This same data set is called the INFILE data set when it is processed by the ZONEIMPORT command.)</p>
<b>Attributes</b>	Sequential; no DCB parameters are required.
<b>Device</b>	Tape or direct access.

---

### PARMLIB Data Set

<b>ddname</b>	PARMLIB.
<b>Use</b>	The PARMLIB data set contains members that define macros and assembler operation codes. During JCLIN processing, SMP/E checks members of SYS1.PARMLIB to determine whether assembler instructions are macro invocations or OPCODEs. SMP/E supplies member GIMOPCDE, which identifies standard assembler OPCODEs. You can add other members to define OPCODEs and macro names not included in GIMOPCDE. For more information about PARMLIB members, see Chapter 33.
<b>Attributes</b>	Partitioned.
<b>Device</b>	Direct access only.

---

## SMPCNTL Data Set

<b>ddname</b>	SMPCNTL.
<b>Use</b>	The SMPCNTL data set contains the SMP/E commands to be processed.
<b>Attributes</b>	Sequential; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB.
<b>Device</b>	Card, tape, direct access, or terminal.

---

## SMPCSI Data Set

<b>ddname</b>	SMPCSI.
<b>Use</b>	The data set specified by the ddname SMPCSI is the CSI containing the global zone. (This data set is also known as the master CSI.) The CSI is the database used by SMP/E to record status and other information about the various target and distribution libraries being supported.
<b>Attributes</b>	VSAM; RECORDSIZE(24 143), KEYS(24 0).
<b>Device</b>	Direct access only.

**Notes:**

1. The low-level qualifier of the data set name must be *CSI*.
2. The format of the records in CSI data sets for SMP/E Release 8.1 is different from the format in SMP/E Release 4, or earlier. Any existing CSI data sets used with SMP/E Release 4, or earlier, that will be used with SMP/E Release 8.1 must, therefore, be converted to SMP/E Release 8.1 format. The SMP/E CONVERT command can be used to do this. See Chapter 5 for more information.
 

**Note:** CSI data sets used with SMP/E Release 5 or higher can be processed using SMP/E Release 8.1 and do not need to be converted.
3. If you have used SMP/E Release 8.1 to update a CSI data set, you might not be able to process that data set with previous releases of SMP/E. For more information, see Appendix H, "Compatibility with Previous SMP/E Releases" on page 865.
4. When running on systems with the required level of Data Facility Product (DFP), SMP/E automatically takes advantage of the local shared resource (LSR) feature of VSAM. This reduces the number of times SMP/E must access data when it is reading CSI data sets. As a result, SMP/E performance is improved for commands such as APPLY, APPLY CHECK, ACCEPT, ACCEPT CHECK, and especially LIST. For more information about this performance improvement, see Appendix I.
5. CSI data sets should usually be allocated dynamically. However, you may want to use the batch local shared resources (BLSR) subsystem with expanded storage hiperspaces (instead of SMP/E's implementation of LSR) to improve SMP/E performance during APPLY and ACCEPT processing for a large number of changes. For more information about BLSR, see Appendix I.

6. For more information about using the EXEC statement to specify the CSI data set containing the global zone, see Appendix B.
7. For information about the target and distribution zones in a CSI data set, see "Zone Statement" on page 624.

---

### SMPDEBUG Data Set

<b>ddname</b>	SMPDEBUG.
<b>Use</b>	The SMPDEBUG data set contains a dump that was requested by the DEBUG command. Depending on the operands specified, it may contain (1) a dump of SMP/E control blocks and storage areas associated with the specified dump points, or (2) a dump of the VSAM RPL control block and additional RPL information for the specified SMP/E function.
<b>Attributes</b>	Sequential; LRECL=121, BLKSIZE=multiple of 121, RECFM=FBA, DISP=MOD.
<b>Device</b>	SYSOUT, printer, direct access, tape, or terminal.

---

### SMPHOLD Data Set

<b>ddname</b>	SMPHOLD.
<b>Use</b>	This may refer to an actual data set, or it may refer to a file on a tape (such as file 4 on an ESO or CUM tape). SMPHOLD contains ++HOLD and ++RELEASE statements to be processed by the RECEIVE command.
<b>Attributes</b>	Sequential; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB.
<b>Device</b>	Direct access or tape.

---

### SMPJCLIN Data Set

<b>ddname</b>	SMPJCLIN.
<b>Use</b>	The SMPJCLIN data set contains a job stream of assembly, link-edit, and copy job steps. This data is typically the stage 1 output from the most recent full or partial system generation, but it may be other data in a similar format, such as output from the SMP/E GENERATE command. This job stream is used as input to the JCLIN command to update or create entries in a target zone.
<b>Attributes</b>	Sequential; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB.
<b>Device</b>	Card, tape, direct access, or terminal.

---

## SMPLIST Data Set

<b>ddname</b>	SMPLIST.
<b>Use</b>	The SMPLIST data set contains the output of all LIST commands.
<b>Attributes</b>	Sequential; LRECL=121, BLKSIZE=multiple of 121, RECFM=FBA, DISP=MOD.
<b>Device</b>	SYSOUT, printer, direct access, tape, or terminal.

**Notes:**

1. If SMPLIST is not defined, all LIST output goes to the SMPOUT data set.
2. If SMPLIST is allocated to a data set, the disposition must be MOD, because SMP/E opens and closes the SMPLIST DD statement at each SET command. If the disposition is SHR or OLD, SMPLIST contains only the output from the last set of commands processed before the end of SMP/E processing.

---

## SMPLOG Data Set

<b>ddname</b>	SMPLOG.
<b>Use</b>	The SMPLOG data set (LOG) contains time-stamped records of SMP/E processing. The records in this data set can be written automatically by SMP/E or added by the user through the LOG command. The data set also contains messages issued by SMP/E, as well as detailed information about data set allocation.
<b>Attributes</b>	Sequential; BLKSIZE=514-32000, RECFM=VB, DISP=MOD. <ul style="list-style-type: none"> <li>• The BLKSIZE value determines how many records are written to the LOG at one time. As a block is filled, it is written to SMPLOG.</li> <li>• If BLKSIZE is less than 514, SMP/E uses the default of 3200.</li> <li>• <b>DISP=MOD</b> must be specified to maintain a cumulative history of SMP/E processing.</li> <li>• SMP/E uses LRECL=510 in allocating the SMPLOG data set.</li> </ul>
<b>Device</b>	Tape or direct access.

**Notes:**

1. Each zone should have its own SMPLOG data set.
2. A secondary log data set (SMPLOGA) should be defined to hold log data when the SMPLOG data set is full. Otherwise, the extra log data is written to the SMPOUT data set, with the date and time stamp encrypted.
3. SMPLOG should be updated only by the LOG command or by processing for other SMP/E commands.
4. Because some messages are longer than in earlier releases of SMP/E, you may need to increase the size of any data sets used for SMP/E messages (such as SMPOUT). How much more space you need depends on the current size of these data sets and which messages are issued. To start, you may want to allocate new data sets twice the size of the old ones.

## SMPLOGA Data Set

<b>ddname</b>	SMPLOGA.
<b>Use</b>	SMPLOGA is a backup LOG data set. If SMPLOGA is defined, it is used automatically when the SMPLOG data set is full. The LOG data set contains time-stamped records of SMP/E processing. The records in this data set can be written automatically by SMP/E or added by the user through the LOG command. The data set also contains messages issued by SMP/E, as well as detailed information about data set allocation.
<b>Attributes</b>	Sequential; BLKSIZE=514-32000, RECFM=VB, DISP=MOD. <ul style="list-style-type: none"><li>• The BLKSIZE value determines how many records are written to the LOG at one time. As a block is filled, it is written to SMPLOG.</li></ul> If BLKSIZE is less than 514, SMP/E uses the default of 3200. The BLKSIZE for the SMPLOGA data set must match the BLKSIZE for the related SMPLOG data set. <ul style="list-style-type: none"><li>• <b>DISP=MOD</b> must be specified to maintain a cumulative history of SMP/E processing.</li><li>• SMP/E uses LRECL=510 in allocating the SMPLOGA data set.</li></ul>
<b>Device</b>	Tape or direct access.

**Notes:**

1. Each zone should have its own SMPLOGA data set.
2. If the SMPLOGA data set is full, the extra log data is written to the SMPOUT data set, with the date and time stamp encrypted.
3. SMPLOGA should be updated only by the LOG command or by processing for other SMP/E commands.
4. Because some messages are longer than in earlier releases of SMP/E, you may need to increase the size of any data sets used for SMP/E messages (such as SMPOUT). How much more space you need depends on the current size of these data sets and which messages are issued. To start, you may want to allocate new data sets twice the size of the old ones.

---

## SMPLTS Data Set

<b>ddname</b>	SMPLTS.
<b>Use</b>	The SMPLTS data set is a target library used to maintain the base version of a load module that specifies a SYSLIB allocation in order to implicitly include modules. A base version of a load module includes only the explicitly defined modules for the load module and it is maintained in the SMPLTS if the load module has been defined to SMP/E with a SYSLIB allocation (its LMOD entry contains a CALLLIBS subentry list). SMP/E uses the load module in the SMPLTS as input when link-editing the load module into its specified target libraries.



**Attributes** Partitioned; RECFM=U, DISP=OLD, BLKSIZE greater than or equal to 1024.

**Device** Direct access only.

**Notes:**

1. Each target zone must have its own SMPLTS data set that is not shared by any other target zone.
2. The SMPLTS data set is required for APPLY, LINK, and RESTORE processing.
3. The BLKSIZE of the SMPLTS data set needs to be 1024 or greater to support load modules that specify the SCTR attribute. A large BLKSIZE is recommended to provide maximum space efficiency.
4. The SMPLTS data set is eligible for RETRY processing, in the same manner as other target libraries, after an x37 ABEND. The size of an SMPLTS data set varies depending on the number of load modules that specify a CALLLIBS sub-entry list in the target zone associated with the SMPLTS.

---

## SMPMTS Data Set

**ddname** SMPMTS.

**Use** The SMPMTS data set (MTS) is a target library for macros that exist only in a distribution library (such as macros in SYS1.AMODGEN). This data set allows the current version of these macros to be used for assemblies during APPLY processing. For more information about MTS entries, see “MTSMAC Entry (SMPMTS)” on page 739.

**Attributes** Partitioned; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB, DISP=OLD.

**Device** Direct access only.

**Notes:**

1. Each target zone must have its own SMPMTS data set that is not shared by any other target zone. This SMPMTS data set may be used with the related distribution zone.
2. For APPLY processing, the SMPMTS data set must be allocated with enough space to hold all the system generation macros, as well as any other macros that do not reside in a target library. This is because functions are now packaged with a complete set of system generation macros.
3. The SMPMTS data set **must** be the first data set in the SYSLIB concatenation for APPLY and RESTORE processing. It can be in the SYSLIB concatenation for ACCEPT processing. For more information on SYSLIB concatenation requirements, see Appendix B.

## SMPOBJ Data Set

<b>ddname</b>	SMPOBJ.
<b>Use</b>	The SMPOBJ data set is used primarily for source-maintained products. It contains preassembled modules that can be used instead of reassembling those modules. These modules must be in load module format—that is, in the same format as modules residing in the distribution library.
<b>Attributes</b>	Partitioned; RECFM=U, DISP=SHR.
<b>Device</b>	Direct access only.

---

## SMPOUT Data Set

<b>ddname</b>	SMPOUT.
<b>Use</b>	The SMPOUT data set contains messages issued during SMP/E processing, as well as dumps of the VSAM RPL, if any dumps were taken. It may also contain LIST output and reports if the SMPLIST and SMPRPT data sets are not defined.
<b>Attributes</b>	Sequential; LRECL=121, BLKSIZE=multiple of 121, RECFM=FBA, DISP=MOD.
<b>Device</b>	SYSOUT, printer, direct access, tape, or terminal.

**Notes:**

1. If SMPOUT is allocated to a data set, the disposition must be MOD, because SMP/E opens and closes the SMPOUT DD statement at each SET command. If the disposition is SHR or OLD, SMPOUT contains only the output from the last set of commands processed before the end of SMP/E processing.
2. Because some messages are longer than in earlier releases of SMP/E, you may need to increase the size of any data sets used for SMP/E messages (such as SMPOUT). How much more space you need depends on the current size of these data sets and which messages are issued. To start, you may want to allocate new data sets twice the size of the old ones.

---

## SMPPTFIN Data Set

<b>ddname</b>	SMPPTFIN.  This ddname can refer to an actual data set or to a file on a tape (such as file 1 on a ESO or CUM tape).
<b>Use</b>	The SMPPTFIN data set contains SYSMODs and ++ASSIGN statements to be processed by the RECEIVE command.
<b>Attributes</b>	Sequential; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB.
<b>Device</b>	Card, tape, direct access, or terminal.

**Note:** If the SMPPTFIN data set is inline, you must make sure that the combination of characters used for the delimiter does not occur in the input itself.

For example, if **DD \*** is specified for SYSIN and the SMPPTFIN data set contains the characters \$\$, then \$\$ cannot be used as the default delimiter for the input or for the delimiter specified on the DLM parameter. (The output of SMP/E service routine GIMDTS contains the characters \$\$.)

---

## SMPPTS Data Set

<b>ddname</b>	SMPPTS.
<b>Use</b>	The SMPPTS data set (PTS) is used as temporary storage for SYSMODs. It contains one member for each SYSMOD that was received. Each member is called an MCS entry and is an exact copy of the SYSMOD as it was received from the SMPPTFIN data set. The name of an MCS entry matches the SYSMOD ID of the SYSMOD it contains. For more information, see "MCS Entry (SMPPTS)" on page 721.
<b>Attributes</b>	<p>Partitioned; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB, DISP=OLD.</p> <p>If you plan to run concurrent jobs, specify <b>DISP=SHR</b> instead of <b>DISP=OLD</b>.</p> <p>The SMPPTS must be large enough to contain all the SYSMODs in the SMPPTFIN data set from which the SYSMODs will be received. This is because during RECEIVE processing, the SMPPTS is used as temporary storage. All the SYSMODs in SMPPTFIN are copied into the SMPPTS, although only the MCS entries for SYSMODs that were actually received are kept.</p> <p>Do not concatenate SMPPTS data sets. SMP/E stores data in the SMPPTS in a particular order. If SMPPTS data sets are concatenated, that order is lost and SMP/E cannot find the data.</p> <p>You <b>must</b> specify a valid data set name for the SMPPTS. <b>NULLFILE</b> and <b>DD DUMMY</b> are invalid for the SMPPTS.</p>
<b>Device</b>	Direct access only.

---

## SMPPUNCH Data Set

<b>ddname</b>	SMPPUNCH.
<b>Use</b>	<p>The SMPPUNCH data set contains output from various SMP/E commands. This output generally consists of commands or control statements.</p> <ul style="list-style-type: none"> <li>• For GENERATE, it contains a job stream for building target libraries.</li> <li>• For REPORT CROSSZONE, it contains commands for installing cross-zone requisites.</li> <li>• For REPORT ERRSYSMODS, it contains commands for installing SYSMODs that resolve the error hold reason IDs for exception SYSMODs.</li> </ul>

- For REPORT SOURCEID, it contains commands for listing SYSMODs associated with the source IDs that were found in the specified zones.
- For REPORT SYSMODS, it contains commands for installing SYSMODs from the input zone that are applicable to the comparison zone.
- For UNLOAD, it contains UCLIN statements for recreating the entries that were unloaded.

**Attributes** Sequential; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB, DISP=MOD.

**Device** Card, tape, or direct access.

**Notes:**

1. If SMPPUNCH is allocated to a data set, the disposition must be MOD, because SMP/E opens and closes the SMPPUNCH DD statement at each SET command. If the disposition is SHR or OLD, SMPPUNCH contains only the output from the last set of commands processed before the end of SMP/E processing.
2. For the UNLOAD command, SMPPUNCH should be allocated to a direct access data set or to tape, because the volume of output is large. You may also want a large BLKSIZE; the larger the BLKSIZE, the fewer the times SMP/E must do I/O.

---

## SMPRPT Data Set

**ddname** SMPRPT.

**Use** The SMPRPT data set contains the reports produced during SMP/E processing.

**Attributes** Sequential; LRECL=121, BLKSIZE=multiple of 121, RECFM=FBA, DISP=MOD.

**Device** SYSOUT, printer, direct access, tape, or terminal.

**Notes:**

1. If SMPRPT is not defined, all report output goes to the SMPOUT data set.
2. If SMPRPT is allocated to a data set, the disposition must be MOD, because SMP/E opens and closes the SMPRPT DD statement at each SET command. If the disposition is SHR or OLD, SMPRPT contains only the reports from the last set of commands processed before the end of SMP/E processing.

---

## SMPSCDS Data Set

**ddname** SMPSCDS.

**Use** The SMPSCDS data set (SCDS) contains backup copies of target zone entries that are created during APPLY processing. These backup copies are made before the entries are (1) changed by inline JCLIN, a ++MOVE MCS, or a ++RENAME MCS, or (2) deleted by an element MCS with the DELETE operand. The

backup copies are used during RESTORE processing to return the entries to the way they were before APPLY processing. For more information about BACKUP entries, see "BACKUP Entries (SMPSCDS)" on page 645.

**Attributes** Partitioned; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB, DISP=OLD.

**Device** Direct access only.

**Notes:**

1. Each target zone must have its own SMPSCDS data set; that data set must be unique to that target zone. This SMPSCDS data set must also be used with the related distribution zone.
2. The format of the records in SMPSCDS data sets for SMP/E Release 8.1 is different from the format in SMP/E Release 4, or earlier. Any existing SMPSCDS data sets used with SMP/E Release 4, or earlier, that will be used with SMP/E Release 8.1 must, therefore, be converted to SMP/E Release 8.1 format. The SMP/E CONVERT command can be used to do this. See Chapter 5 for more information.

**Note:** SCDS data sets used with SMP/E Release 5, or later, can be processed by SMP/E Release 8.1 and do not need to be converted.

---

## SMPSNAP Data Set

**ddname** SMPSNAP.

**Use** The SMPSNAP data set is used for snap dump output. When a severe error occurs, such as an abend or severe VSAM return code, SMP/E requests a snap dump of its storage before doing any error recovery. In addition, the DEBUG command may request a snap dump of SMP/E storage when specified messages are issued, or it may request a snap dump of control blocks and storage areas associated with a specified dump point.

**Attributes** Sequential.

**Device** SYSOUT, printer, direct access, tape, or terminal.

---

## SMPSTS Data Set

**ddname** SMPSTS.

**Use** The SMPSTS data set (STS) is a target library for source that exists only in a distribution library. This data set allows the current version of these modules to be used for assemblies during APPLY processing.

**Attributes** Partitioned; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB, DISP=OLD.

**Device** Direct access only.

**Note:** Each target zone must have its own SMPSTS data set, which may not be shared by any other target zone. This SMPSTS data set can also be used with the related distribution zone.

---

## SMPTLIB Data Set

**ddname** SMPTLIB.

**Use** SMPTLIB data sets (TLIBs) are used as temporary storage for relative files that are loaded from SMPPTFIN during RECEIVE processing. They are deleted when the associated SYSMOD is deleted by REJECT, RESTORE, or ACCEPT processing.

You can have SMP/E dynamically allocate the TLIB data sets, or you can allocate them yourself before RECEIVE processing. For information about how SMP/E allocates the TLIBs, see “Receiving SYSMODs Packaged in Relative Files” on page 243. (Regardless of how the SMPTLIB data sets are allocated, they do not appear in the File Allocation report.)

**Notes:**

1. No DD statement or DDDEF entry is required for the SMPTLIB data sets if they are preallocated and cataloged.
2. If you allocate and catalog the SMPTLIB data sets yourself, make sure they are allocated on the volume specified in the catalog.
3. If you allocate the SMPTLIB data sets yourself but do not catalog them, make sure they are allocated on the volume specified in the SMPTLIB DD statement or DDDEF entry being used for the command you are processing.
4. If you need to specify a unit that is not in SYSALLDA, and the unit is not set by use of a DATACLAS or DATACLAS filter routine, you must use a DDDEF entry instead of a DD statement to allocate the SMPTLIB data sets.
5. If you are using SMS to manage your data sets, you can set up the unit, volume, and space allocation through a DATACLAS or DATACLAS filter routine instead of specifying them on a DD statement or DDDEF entry.
6. SMPTLIB data sets should not be allocated as PDSEs, because IEBCOPY does not support copying an unloaded PDS load library from tape to a PDSE load library on DASD.

**Attributes** Partitioned.

The SMPTLIB definition must specify at least one volume with enough space for SMP/E to dynamically allocate storage for the libraries. You can specify up to five volumes in the DDDEF entry or on the DD statement.

If you are using MSS volumes, the UNIT parameter must specify either the number of volumes, the unit count, or P (meaning one unit per volume specified).

Here are two examples of SMPTLIB DD statements:

```
//SMPTLIB DD UNIT=3350,VOL=SER=335001
//SMPTLIB DD UNIT=3350,
//          VOL=SER=(335001,335002)
```

If you use DDDEFs to have SMP/E dynamically allocate the SMPTLIB data sets, you cannot specify the initial or final disposition. SMP/E determines the disposition based on the command it is processing. Table 28 on page 619 shows these commands and the dispositions used.

**Device** Direct access only.

Command	Conditions	Disposition
ACCEPT	<b>NOPURGE</b> is not specified in the OPTIONS entry that is in effect.	(OLD,DELETE)
	<b>NOPURGE</b> is specified in the OPTIONS entry that is in effect.	(SHR,KEEP)
APPLY	–	(SHR,KEEP)
RECEIVE	SMPTLIBs are neither preallocated nor cataloged.	(NEW,CATALOG)
	SMPTLIBs are preallocated and may or may not be cataloged.	(OLD,CATALOG)
REJECT	–	(OLD,DELETE)
RESTORE	<b>NOREJECT</b> is not specified in the OPTIONS entry that is in effect.	(OLD,DELETE)
	<b>NOREJECT</b> is specified in the OPTIONS entry that is in effect.	(OLD,KEEP)

## SMPWRK1 Data Set

**ddname** SMPWRK1.

**Use** The SMPWRK1 data set is used as temporary storage for macro updates and replacements that will be processed by the IEBUPDTE and IEBCOPY programs. During APPLY and ACCEPT processing, SMP/E places the input in this data set before calling the utility.

**Attributes** Partitioned; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB, DISP=(NEW,DELETE).

Specifying **DISP=(NEW,DELETE)** minimizes space loss problems. SMPWRK1 is generally needed only for the duration of the SMP/E job step. To keep the data set longer than that, you must use a different DISP value and compress the data set to reclaim space.

The SMS-related options DATACLAS, MGMTCLAS, STORCLAS, and DSNTYPE provide additional data set support. For more information, refer to the appropriate data facilities storage management manuals.

**Device** Direct access only.

---

## SMPWRK2 Data Set

<b>ddname</b>	SMPWRK2.
<b>Use</b>	The SMPWRK2 data set is used as temporary storage for source updates and source replacements that will be processed by the IEBUPDTE and IEBCOPY programs. During APPLY and ACCEPT processing, SMP/E places the input in this data set before calling the utility.
<b>Attributes</b>	<p>Partitioned; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB, DISP=(NEW,DELETE).</p> <p>Specifying <b>DISP=(NEW,DELETE)</b> minimizes space loss problems. SMPWRK2 is generally needed only for the duration of the SMP/E job step. To keep the data set longer than that, you must use a different DISP value and compress the data set to reclaim space.</p> <p>The SMS-related options DATACLAS, MGMTCLAS, STORCLAS, and DSNTYPE provide additional data set support. For more information, refer to the appropriate data facilities storage management manuals.</p>
<b>Device</b>	Direct access only.

---

## SMPWRK3 Data Set

<b>ddname</b>	SMPWRK3.
<b>Use</b>	The SMPWRK3 data set is used as temporary storage for object modules supplied by a SYSMOD, object modules created by assemblies, and IMASPZAP control cards following ++ZAP statements.
<b>Attributes</b>	<p>Partitioned; LRECL=80, BLKSIZE=multiple of 80 (maximum=3200), RECFM=FB, DISP=(NEW,DELETE).</p> <p>Specifying <b>DISP=(NEW,DELETE)</b> minimizes space loss problems. SMPWRK3 is generally needed only for the duration of the SMP/E job step. To keep the data set longer than that, you must use a different DISP value and compress the data set to reclaim space.</p> <p>If you want to save assembled object modules until SMP/E has successfully applied or accepted the SYSMODs that caused the assemblies, specify <b>DISP=(NEW,KEEP)</b>. This allows SMP/E to reuse the assembled object modules if the APPLY or ACCEPT command fails. For more information, see “Reusing Previous Assemblies” on page 45 in the ACCEPT chapter and “Reusing Previous Assemblies” on page 97 in the APPLY chapter.</p> <p><b>Note:</b> If SMPWRK3 is a permanent data set, make sure to specify <b>OLD</b> as the initial disposition. Do not specify <b>SHR</b>. An initial disposition of SHR may cause an abend.</p> <p>The SMS-related options DATACLAS, MGMTCLAS, STORCLAS, and DSNTYPE provide additional data set support. For more information, refer to the appropriate data facilities storage management manuals.</p>



---

<b>Device</b>	Direct access only
---------------	--------------------

---

### SMPWRK4 Data Set

<b>ddname</b>	SMPWRK4.
<b>Use</b>	The SMPWRK4 data set is used as temporary storage for IMASPZAP and link-edit input containing EXPAND control statements.
<b>Attributes</b>	<p>Partitioned; LRECL=80, BLKSIZE=multiple of 80 (maximum=3200), RECFM=FB, DISP=(NEW,DELETE).</p> <p>Specifying <b>DISP=(NEW,DELETE)</b> minimizes space loss problems. SMPWRK4 is generally needed only for the duration of the SMP/E job step. To keep the data set longer than that, you must use a different DISP value and compress the data set to reclaim space.</p> <p>The SMS-related options DATACLAS, MGMTCLAS, STORCLAS, and DSNTYPE provide additional data set support. For more information, refer to the appropriate data facilities storage management manuals.</p>
<b>Device</b>	Direct access only.

---

### SMPWRK6 Data Set

<b>ddname</b>	SMPWRK6.
<b>Use</b>	The SMPWRK6 data set is used during APPLY and ACCEPT processing as temporary storage for inline replacements for data elements. SMP/E places the input in this data set so it can be directly accessed and installed by the copy utility or SMP/E.
<b>Attributes</b>	<p>Partitioned; LRECL=80, BLKSIZE=multiple of 80 (default=3200), RECFM=FB, DISP=(NEW,DELETE).</p> <p>Specifying <b>DISP=(NEW,DELETE)</b> minimizes space loss problems. SMPWRK6 is generally needed only for the duration of the SMP/E job step. To keep the data set longer than that, you must use a different DISP value and compress the data set to reclaim space.</p> <p>The SMS-related options DATACLAS, MGMTCLAS, STORCLAS, and DSNTYPE provide additional data set support. For more information, refer to the appropriate data facilities storage management manuals.</p>
<b>Device</b>	Direct access only.

---

### SMPnnnnn Data Set

<b>ddname</b>	SMPnnnnn, where <i>nnnnn</i> is a number from 00000 through 99999.
<b>Use</b>	SMP/E allocates SMPnnnnn data sets for its own internal processing. To avoid processing errors, do not assign such a ddname to any data set.

---

**SYSLIB Data Set**

<b>ddname</b>	SYSLIB.
<b>Use</b>	<p>The SYSLIB data set is a concatenation of macro libraries that are to be used by the assembler utility.</p> <p>For APPLY and RESTORE processing, the data sets should be concatenated in this order:</p> <ol style="list-style-type: none"><li>1. SMPMTS</li><li>2. MACLIB</li><li>3. MODGEN</li><li>4. Target system macro libraries (such as libraries specified for SYSLIB on the ++MAC statement)</li><li>5. Distribution macro libraries (such as libraries specified for DISTLIB on the ++MAC statement)</li></ol> <p>For more information about the proper SYSLIB concatenation for ACCEPT processing, see Appendix B.</p>
<b>Attributes</b>	Partitioned; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB.
<b>Device</b>	Direct access only.

---

**SYSPRINT Data Set**

<b>ddname</b>	SYSPRINT.
<b>Use</b>	<p>You can specify different SYSPRINT data sets for each of the utilities that SMP/E calls. This can be done with the PRINT value in the appropriate UTILITY entry. For more information, see "UTILITY Entry (Global Zone)" on page 787.</p>
<b>Attributes</b>	<p>Sequential; DISP=MOD.</p> <p>Do not specify LRECL, BLKSIZE, or RECFM unless they are compatible with the attributes used by the utilities called.</p> <p>If SYSPRINT is allocated to a data set, the disposition must be MOD, because SMP/E opens and closes the SYSPRINT DD statement at each SET command. If the disposition is SHR or OLD, SYSPRINT contains only the output from the last set of commands processed before the end of SMP/E processing.</p>
<b>Device</b>	<p>SYSOUT, printer, direct access, or tape.</p> <p>SYSOUT or a tape is recommended, because SYSPRINT might be opened with different DCB attributes by the various utilities and service aids called by SMP/E.</p>
<b>Note:</b>	<p>How you specify the SYSPRINT data set can affect whether a listing of the utility output is produced. For example, no listing is produced if the PRINT value in the UTILITY entry specifies either of the following:</p> <ul style="list-style-type: none"><li>• A DDDEF for a DUMMY data set</li></ul>

- A DDDEF for a data set that is sent to a SYSOUT class that suppresses output

---

## SYSPUNCH Data Set

<b>ddname</b>	SYSPUNCH.
<b>Use</b>	The SYSPUNCH data set is a temporary data set containing object modules assembled by running the job stream produced by system generation or the GENERATE command. These modules are not installed in the distribution libraries at ACCEPT time.
<b>Attributes</b>	Partitioned; LRECL=80, BLKSIZE=multiple of 80, RECFM=FB. Do not specify a DISP value. Instead, let SMP/E use its defaults: <ul style="list-style-type: none"><li>• DISP=(NEW,PASS) for the first job generated</li><li>• DISP=(OLD,PASS) for any subsequent jobs</li></ul>
<b>Device</b>	Direct access only.

---

## SYSUT1, SYSUT2, and SYSUT3 Data Sets

<b>ddname</b>	SYSUT1, SYSUT2, and SYSUT3.
<b>Use</b>	These are scratch data sets for SMP/E and the utilities it calls. They can be used instead of the following data sets when certain utilities are called: <ul style="list-style-type: none"><li>• SYSIN<ul style="list-style-type: none"><li>– For invocations of the copy utility</li><li>– For some invocations of update utility</li><li>– For some invocations of the x37 REPLY COMPRESS utility</li></ul></li><li>• SYSLIN for invocations of the link-edit utility</li><li>• SYSUT2 for invocations of the assembler utility</li></ul> SYSUT1 and SYSUT2 are also used by the GIMDTS service routine: <ul style="list-style-type: none"><li>• SYSUT1 points to the data set containing the input in its original format.</li><li>• SYSUT2 points to the data set containing the transformed output.</li></ul>
<b>Attributes</b>	Sequential; DISP=(NEW,DELETE)
<b>Device</b>	Direct access only.

## SYSUT4 Data Set

<b>ddname</b>	SYSUT4.
<b>Use</b>	This data set can be used instead of the SYSIN data sets when certain utilities are called: <ul style="list-style-type: none"><li>• For invocations of the x37 RETRY COMPRESS utility</li><li>• For some invocations of the assembler utility</li></ul>
<b>Attributes</b>	Sequential; DISP=(NEW,DELETE)
<b>Device</b>	Direct access only.

---

## Target Library

<b>ddname</b>	The ddname for a target library should match the low-level qualifier of the data set name. For example, the ddname for SYS1.LINKLIB should be LINKLIB.
<b>Use</b>	Target libraries contain updated versions of macros, source modules, and load modules that were stored during APPLY or RESTORE processing. They are the libraries used for your running system. You must provide a DDDEF entry or DD statement for each target library that is being processed.
<b>Attributes</b>	Partitioned.
<b>Device</b>	Direct access only.

---

## Text Library (TXLIB)

<b>ddname</b>	The ddname for a text library must match the TXLIB value on the ++JCLIN or element MCS. For example, the ddname for the text library on statement ++MOD(MODA) TXLIB(LIBX) must be LIBX.
<b>Use</b>	Text libraries contain JCLIN input or replacements for macros, source, or object modules that have not been link-edited. They are used when the JCLIN or elements are provided in partitioned data sets rather than inline or in relative files.
<b>Attributes</b>	Partitioned.
<b>Device</b>	Direct access only.

---

## Zone Statement

<b>ddname</b>	The ddname must match the name of the target or distribution zone.
<b>Use</b>	The <i>zone</i> DD statement is used by SMP/E to access an individual target or distribution zone in a CSI data set. For example, to have SMP/E access target zone MVSTGT1, you can provide a DD statement like this one: <pre>//MVSTGT1 DD DSN=SMPE.SMPCSI.CSI,DISP=SHR</pre>

If you do not provide a DD statement for a target or distribution zone, SMP/E allocates the zone dynamically using the ZONEINDEX information in the GLOBALZONE entry.

**Attributes** VSAM; RECORDSIZE(24 143), KEYS(24 0).

**Device** Direct access only.

**Notes:**

1. The low-level qualifier of the data set name must be *CSI*.
2. The format of the records in CSI data sets for SMP/E Release 8.1 is different from the format in SMP/E Release 4, or earlier. Any existing CSI data sets used with SMP/E Release 4, or earlier, that will be used with SMP/E Release 8.1 must, therefore, be converted to SMP/E Release 8.1 format. The SMP/E CONVERT command can be used to do this. See Chapter 5 for more information.  
**Note:** CSI data sets used with SMP/E Release 5, or higher, can be processed by SMP/E Release 8.1 and do not need to be converted.
3. If you have used SMP/E Release 8.1 to update a zone, you might not be able to process that zone with previous releases of SMP/E. For more information, see Appendix H, "Compatibility with Previous SMP/E Releases" on page 865.
4. When running on systems with the required level of DFP, SMP/E automatically takes advantage of the local shared resource (LSR) feature of VSAM. This reduces the number of times SMP/E must access data when it is reading CSI data sets. As a result, SMP/E performance is improved for commands such as APPLY, APPLY CHECK, ACCEPT, ACCEPT CHECK, and especially LIST. For more information about this performance improvement, see Appendix I.
5. CSI data sets should usually be allocated dynamically. However, you may want to use the batch local shared resources (BLSR) subsystem with expanded storage hiperspaces (instead of SMP/E's implementation of LSR) to improve SMP/E performance during APPLY and ACCEPT processing for a large number of changes. For more information about BLSR, see Appendix I.
6. For information about the CSI data set containing the global zone, see "SMPCSI Data Set" on page 609.



---

## Chapter 35. SMP/E Data Set Entries

This chapter describes the entries in the various data sets SMP/E uses. It discusses the following:

- How these data sets are organized
- How the entries in these data sets are organized
- How to create, update, and obtain information about these entries

**Note:** The syntax rules for the UCLIN, LIST, and UNLOAD commands used to process these entries are described in the individual chapters on these commands.

Table 29 shows where to find a detailed description of each entry.

*Table 29. Index to Descriptions of SMP/E Entries*

<b>Zone Type or Data Set</b>	<b>Entry Type</b>	<b>Page</b>
Distribution and target zone	ASSEM	640
	Data element	648
	DDDEF	654
	DLIB	670
	DLIBZONE	675
	HFS	687
	LMOD	698
	MAC	715
	MOD	723
	SRC	750
SYSMOD	759	
TARGETZONE	782	
Global zone	DDDEF	654
	FMIDSET	679
	GLOBALZONE	682
	HOLDDATA	695
	OPTIONS	741
	SYSMOD	774
	UTILITY	787
ZONESET	794	
SMPPTS	MCS	721
SMPSCDS	BACKUP	645
SMPMTS	MTSMAC	739
SMPSTS	STSSRC	757

---

### How the Data Sets Are Organized

SMP/E uses the following data sets as a database for its processing: the CSI, PTS, SCDS, MTS, and STS. It is important to understand how SMP/E uses each of these data sets and how they are related.

The following is a description of each data set:

- The **CSI** data set is a VSAM data set that serves as the primary data set for SMP/E. SMP/E divides the CSI into multiple partitions using the VSAM key structure. Each partition is referred to as a *zone*.

There are three types of zones:

1. **Global zone.** A single global zone is used to record information about SYSMODs that have been received. The global zone also contains information enabling SMP/E to access the other two types of zones, and information allowing you to tailor parts of SMP/E processing.
2. **Target zones.** One or more target zones are used to record information about the status and structure of the operating system (or target) libraries. These zones serve the same purpose as the CDS and CRQ in SMP4.
3. **Distribution zones.** One or more distribution zones are used to record information about the status and structure of the distribution libraries (or DLIBs). These zones serve the same purpose as the ACDS and ACRQ in SMP4.

In Figure 86 on page 629 the CSI is pictured as one data set. In fact, you can group all your zones within one VSAM data set, or divide them up into multiple VSAM data sets, even one zone per data set. The choice is yours; it is based on such factors as:

- Which packs the associated target and distribution libraries are on. It is advisable to keep the associated zone on the same pack as the libraries controlled from the zone, so when the pack is dumped, the data and description are kept synchronized.
- The organization of your shop. Totally separate service organizations for different products may require many separate data sets.

Figure 87 on page 630 illustrates the same zone relationship as Figure 86 on page 629, with the CSI spread across multiple VSAM data sets.

- The **PTS** is used strictly as a storage data set for SYSMODs. A SYSMOD is read from the SMPPTFIN data set and stored directly on the PTS without any modifications or SMP/E information. It is, therefore, related to the global zone in that both data sets contain information about the received SYSMODs. You can, therefore, look at the global zone and the PTS as a pair of data sets that must be processed (such as scratched, saved, or modified) concurrently.
- The **SCDS** is used by SMP/E to store backup copies of modified target zone entries during apply processing of a SYSMOD with inline JCLIN. Thus, the SCDS is directly related to a specific target zone, and each target zone must have its own SCDS.
- The **MTS** is a library in which SMP/E stores updated copies of macros during apply when no other target macro library is identified. The MTS is, therefore, related to a specific target zone, and each target zone must have its own MTS data set.
- The **STS** is a library in which SMP/E stores updated copies of source during APPLY when no other target source library is identified. The STS is, therefore, related to a specific target zone, and each target zone must have its own STS data set.



The relationships among SMP/E data sets and zones are shown in Figure 86 on page 629 for a single CSI data set, and in Figure 87 on page 630 for multiple CSI data sets.

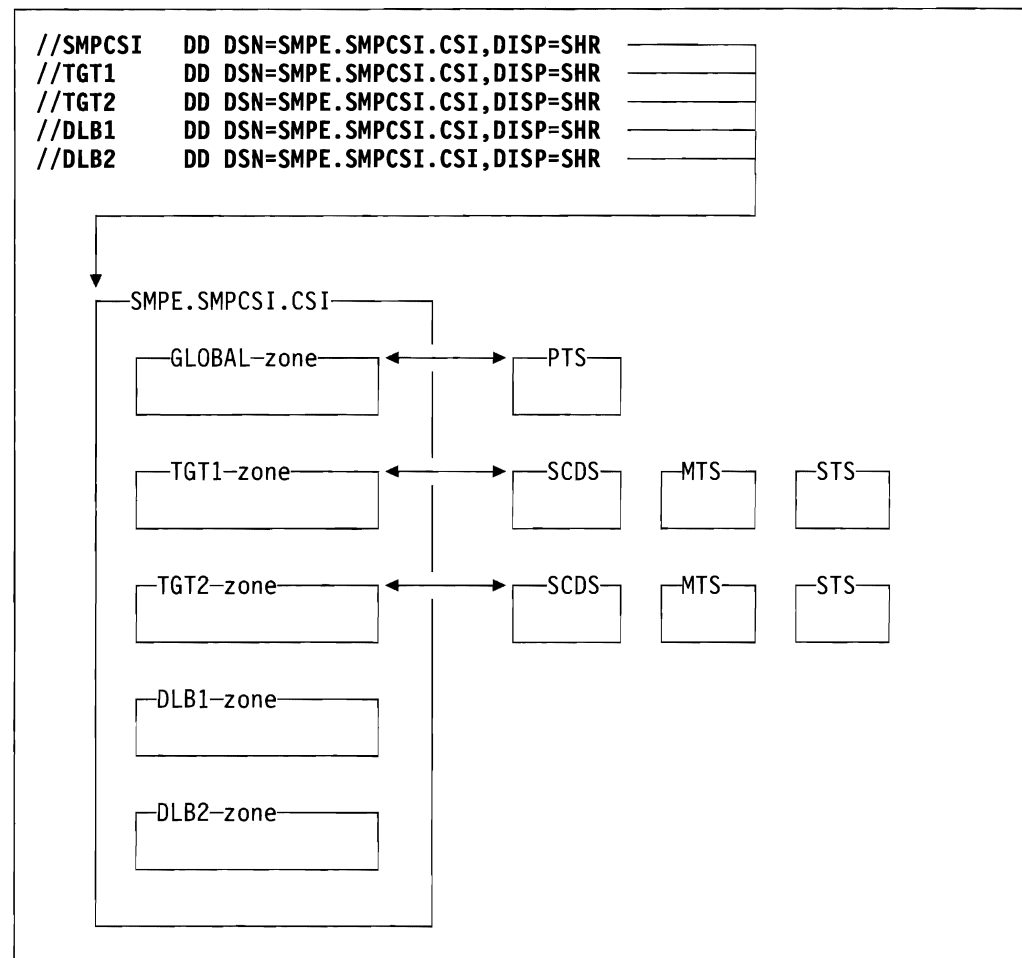


Figure 86. Single-CSI Structure

## How Data Set Entries Are Organized

Within the global zone, target zone, and distribution zone, SMP/E keeps many types of entries. These can be divided into two categories:

- Those that are used to control SMP/E processing. These consist of:
  - GLOBALZONE definition entry
  - TARGETZONE definition entry
  - DLIBZONE definition entry
  - OPTIONS entries
  - UTILITY entries
  - DDDEF entries
  - FMIDSET entries
  - ZONESET entries

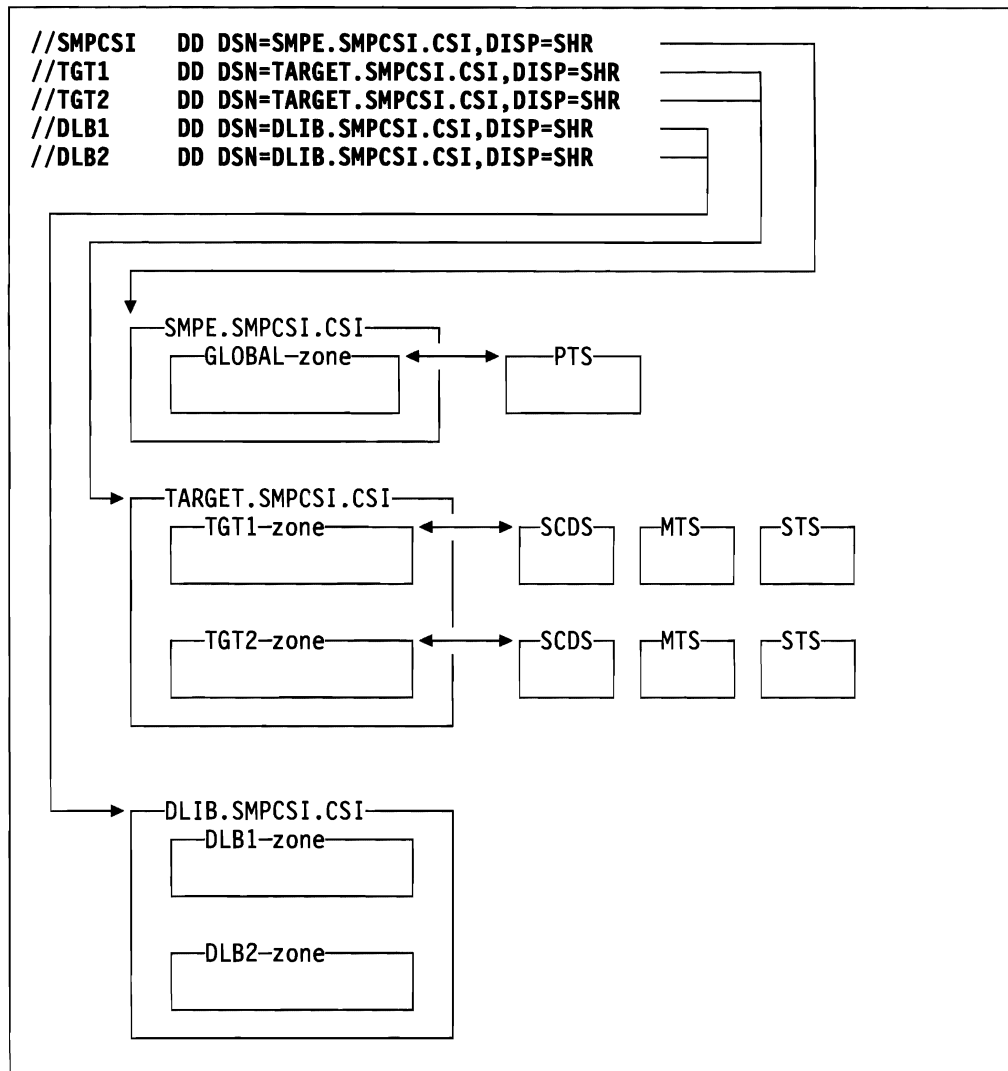


Figure 87. Multiple-CSI Structure

- Those that are used to define the status and structure of the target libraries and distribution libraries. These consist of:
  - ASSEM entries
  - Data element entries
  - DLIB entries
  - HFS entries
  - LMOD entries
  - MAC entries
  - MOD entries
  - SRC entries
  - SYSMOD entries

**Note:** The SYSMOD entries also contain information referred to as HOLDDATA.

The following sections describe the entry relationships for the entries that control processing, and entries that define status and structure.

## Entries that Control Processing

The starting point for all SMP/E processing is the SMPCSI DD statement. This DD statement directs SMP/E to the CSI data set containing the **GLOBALZONE** entry. The next step after obtaining the GLOBALZONE entry depends on which zone you direct SMP/E to process. (For more information on identifying which zone SMP/E is to process, see Chapter 21.)

### Processing the Global Zone

Once the GLOBALZONE entry has been obtained and you direct SMP/E to process the global zone (for instance to receive SYSMODs), the OPTIONS subentry in the GLOBALZONE entry directs SMP/E to the correct **OPTIONS** entry to use. The OPTIONS entry contains information about the SMP/E processing options to be used. During processing, SMP/E may have to invoke one of the system utilities. Another entry, the **UTILITY** entry, is used to define to SMP/E information about the utility program to invoke, the parameters to pass to it, and the return code to expect from it. SMP/E finds the UTILITY entry through the OPTIONS entry that names the UTILITY entry. Thus, for SMP/E to use the correct utility program, you must define **both** the UTILITY entry that describes the utility program and the OPTIONS entry that names that UTILITY entry.

The other processing entries in the global zone are the DDDEF, FMIDSET, and ZONESET entries.

- **DDDEF entries** provide information used by SMP/E to dynamically allocate data sets. These entries are not connected to any other processing entry. They are accessed individually, by name, as SMP/E needs information to dynamically allocate a specific data set. They are not shown in Figure 88 on page 632.
- **FMIDSET entries** define sets of FMIDs. Various other commands, such as APPLY and ACCEPT, can then process SYSMODs by FMIDSET.
- **ZONESET entries** define sets of zones. The REPORT command can then check for SYSMODs in the zones defined in the ZONESET.

Each of these entries (GLOBALZONE, OPTIONS, UTILITY, DDDEF, FMIDSET, and ZONESET) exists in the global zone.

Figure 88 on page 632 illustrates how the various global zone processing entries are related to one another. A more detailed explanation of each entry, and the data contained within it, can be found in subsequent sections of this chapter.

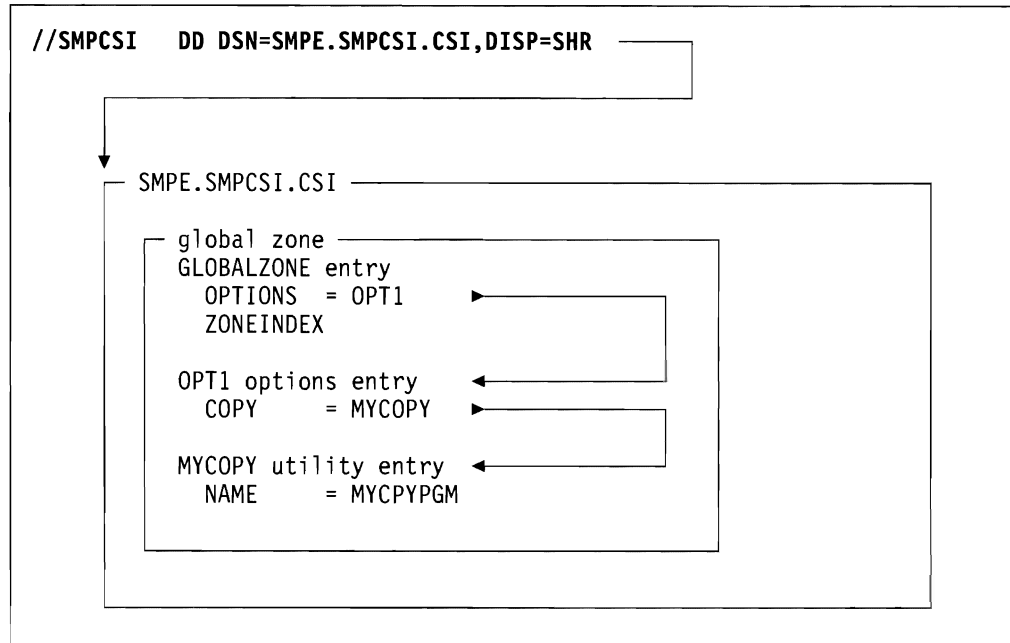


Figure 88. Global Zone: Relationships between Entries that Control Processing

### Processing a Target Zone or a Distribution Zone

A target zone and a distribution zone are processed in very similar ways. The only difference is the name of the controlling entry. Therefore, they are covered together, and the differences are pointed out as appropriate.

When you direct SMP/E to process a target zone (for instance to apply a PTF) or a distribution zone (for instance to accept that PTF), SMP/E accesses the **ZONEINDEX subentries** in the GLOBALZONE entry. Those subentries list the target zones and distribution zones that you have defined, including the zone type, target or DLIB, and the name of the CSI data set on which they reside.

After SMP/E has determined that the zone specified is valid, it uses the CSI data set name specified to dynamically allocate a DD statement to access the required data set (optionally, SMP/E looks for a DD statement equal to the zone name). SMP/E can now access that DD statement to obtain further processing information for the zone.

The first entry accessed is the **TARGETZONE entry** (for target zones) or the **DLIBZONE entry** (for distribution zones). Each of these, in turn, directs SMP/E to the correct **OPTIONS entry** to use, which in turn directs SMP/E to the correct **UTILITY entries**. Target and distribution zones also contain DDDEF entries.

- The **OPTIONS and UTILITY entries** serve the same purpose for the target zone and distribution zone as for the global zone. However, the **OPTIONS and UTILITY entries** used to process a target zone or distribution zone are defined not in the target or distribution zone, but in the global zone that points to the target or distribution zone.
- **TARGETZONE and DLIBZONE entries** contain a **RELATED subentry**, which identifies a related zone. For a target zone, the **RELATED subentry** identifies the distribution zone from which this target zone was built. For a distribution

zone, the RELATED subentry identifies the target zone that was built from these distribution libraries.

- **DDDEF entries** in the target zone and distribution zone are not connected to any other entries. They provide information used by SMP/E to dynamically allocate data sets.

Figure 89 on page 634 shows how the various target zone, distribution zone, and global zone processing entries are related. A more detailed explanation of each entry, and the data contained within it, can be found in subsequent sections of this chapter. Because neither the FMIDSET entries nor the DDDEF entries are connected to any other distribution zone or target zone entry, they are not shown in this figure.

## Entries that Define Status and Structure

Once the processing control information for the specified zone has been determined, SMP/E uses the status and structure entries within the various zones to determine what should be done.

### Processing the Global Zone

Processing the global zone is fairly simple, in that only one structure and status entry exists, SYSMOD. **SYSMOD entries** are used to determine whether a SYSMOD has been received already. Various indicators within the SYSMOD entry also indicate whether the SYSMOD has been applied or accepted. The SYSMOD entries, although not directly connected to the PTS **MCS entries**, are implicitly connected, in that SMP/E assumes that there is a one-to-one relationship between global zone SYSMOD entries and PTS MCS entries.

### Processing the Target Zone

Processing the target zone is much more complex than processing the global zone, because the operations performed are much more complex and require more entry types to direct SMP/E. The primary purpose of the target zone entries is to enable SMP/E to apply new function and service to the target system libraries; thus, the APPLY command is used to describe the relationship between the various target zone entries.

The starting point for applying a SYSMOD is actually the **global zone SYSMOD entries**. From the global zone SYSMOD entry, SMP/E obtains information about the relationship between this SYSMOD and other SYSMODs. SMP/E then uses other global zone SYSMOD entries and the **target zone SYSMOD entries** to resolve these relationships.

Once the eligible SYSMODs have been determined, the actual MCS statements from the PTS are accessed and lead SMP/E to the other target zone entries:

- ++MAC and ++MACUPD statements lead to target zone MAC entries.
- ++SRC and ++SRCUPD statements lead to target zone SRC entries.
- ++MOD and ++ZAP statements lead to target zone MOD and LMOD entries.
- Data element statements lead to target zone data element entries.
- ++HFS statements lead to target zone HFS entries.

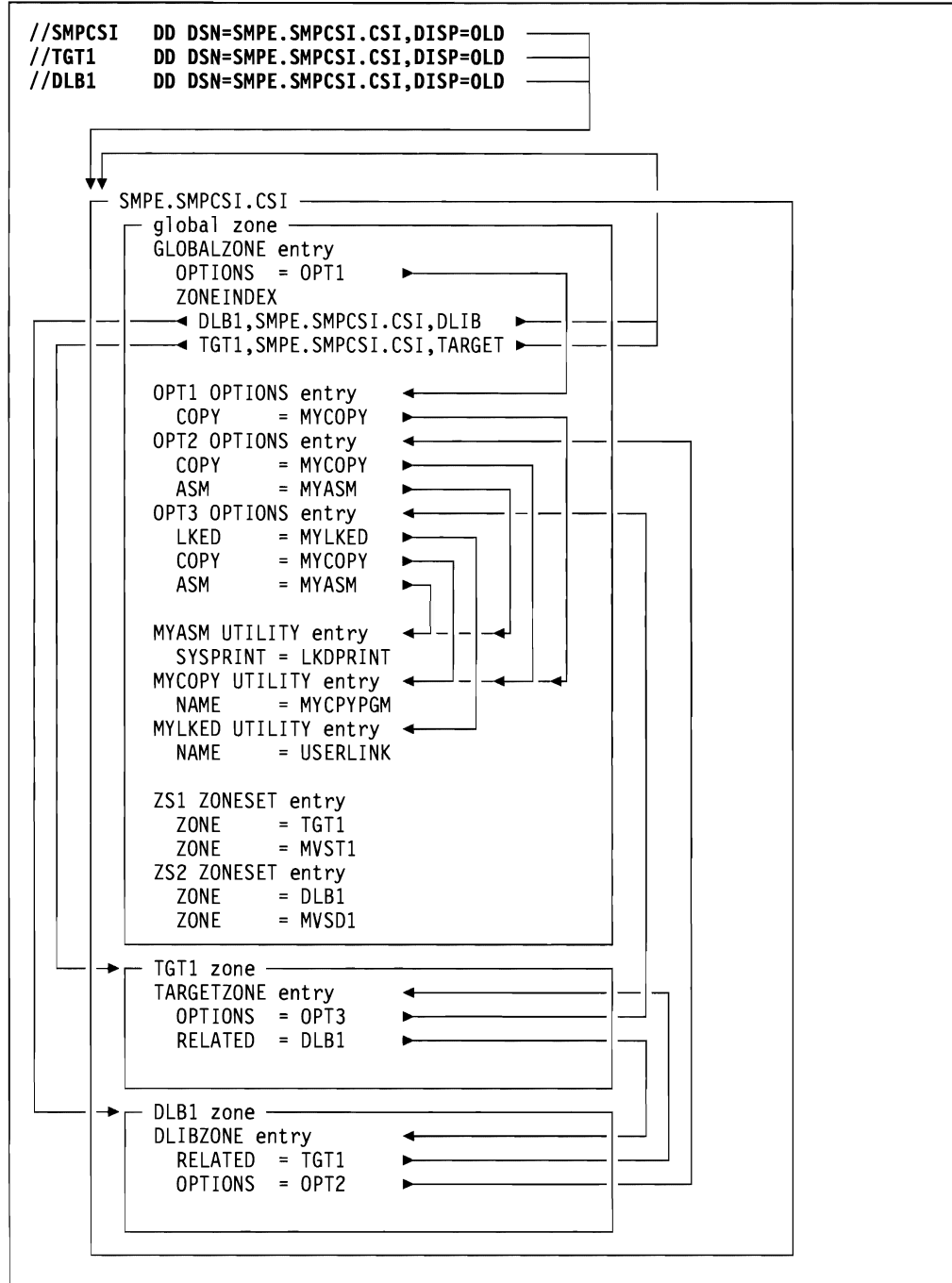


Figure 89. Target Zone and Distribution Zone: Relationships between Entries that Control Processing

If the SYSMOD replaces or modifies a macro (++MAC or ++MACUPD in SYSMOD), SMP/E uses the target zone **MAC entry** to determine the functional and service level of the macro and to determine whether any assemblies must be redone as a result of the macro update. This is done by checking the GENASM subentries in the target zone MAC entry. The GENASM subentries contain the names of either ASSEM entries or SRC entries in the target zone. In either case, SMP/E accesses the appropriate entry and uses the information stored there to perform the required assemblies.

If the SYSMOD replaces or modifies source (++SRC or ++SRCUPD in SYSMOD), SMP/E uses the target zone **SRC entry** to determine the functional and service level of the source and the library containing the source code, and then performs the required assembly.

After an assembly has been done (either using data from the ASSEM entry or SRC entry), SMP/E knows that the resulting object deck must be link-edited into the target libraries somewhere. Although there is no direct connection (that is, no sub-entry value present) between the ASSEM or SRC entries and a **MOD entry**, there is an implicit relationship. SMP/E assumes that for each ASSEM and SRC entry, there exists a MOD entry with the same name. Thus, after performing the assembly, SMP/E accesses the corresponding MOD entry to determine where to install the object deck. This leads us to the same point as if a MOD were supplied in the SYSMOD.

If the SYSMOD replaces or modifies a module (++MOD or ++ZAP in SYSMOD), or if an assembly was done for a SRC or ASSEM entry, SMP/E uses the target zone **MOD entry** to determine the functional and service level of the module and the load modules into which the module should be linked. Information about load modules is kept in the LMOD subentries in the MOD entry.

The LMOD subentries within the MOD entry lead SMP/E to the target zone **LMOD entries**. These represent load modules that exist in the target libraries. LMOD entries contain all the information necessary either to relink the load module or to superzap it.

If the SYSMOD replaces a data element (a data element MCS is in the SYSMOD), SMP/E uses the target zone **data element entry** to determine the functional and service level of the data element. It then gets the data element installed in the appropriate target library.

If the SYSMOD replaces an HFS element (a ++HFS MCS is in the SYSMOD), SMP/E uses the target zone **HFS entry** to determine the functional and service level of the HFS element. It then gets the HFS element installed in the appropriate target library (which is actually in a hierarchical file system).

After all the updating, assembling, linking, and so on, are done, you arrive back almost at the starting point, the SYSMOD entry. The difference is that this is the target zone SYSMOD entry rather than the global zone SYSMOD entry. The target zone SYSMOD entry contains all the information about what has been done as a result of installing this SYSMOD.

That leaves us with one target zone entry not discussed, the **DLIB entry**. The DLIB entries are not connected to any other entry. They are used by SMP/E to keep information about libraries that are totally copied during product installation, and are used during APPLY processing to create the appropriate element and LMOD entries for elements coming from the copied DLIB.

Figure 90 on page 636 shows how the various SMP/E status and structure entries are related. A more detailed explanation of each entry, and the data contained within it, can be found in subsequent sections of this chapter.





## Processing the Distribution Zone

Processing the distribution zone is similar to processing the target zone, but more complex than processing the global zone. The primary purpose of the distribution zone entries is to enable SMP/E to accept new function and service to the distribution libraries; it is also used to remove a SYSMOD from the system. The ACCEPT command will be used to describe the relationship between the various distribution zone entries.

The starting point for accepting a SYSMOD is actually the **global zone SYSMOD entry**. From that entry, SMP/E obtains information about the relationship between this SYSMOD and other SYSMODs. SMP/E then uses other global zone SYSMOD entries and the **distribution zone SYSMOD entries** to resolve these relationships. In addition, SMP/E checks the target zone SYSMOD entries to ensure that the selected SYSMODs have been applied.

After the eligible SYSMODs have been determined, the various MCS statements lead SMP/E to the other distribution zone entries:

- ++MAC and ++MACUPD statements lead to distribution zone MAC entries.
- ++SRC and ++SRCUPD statements lead to distribution zone SRC entries.
- ++MOD and ++ZAP statements lead to distribution zone MOD entries.
- Data element statements lead to distribution zone data element entries.
- ++HFS statements lead to distribution zone HFS entries.

If the SYSMOD replaces or modifies a macro (++MAC or ++MACUPD in SYSMOD), SMP/E uses the distribution zone **MAC entry** to determine the functional and service level of the macro and to determine whether any assemblies must be redone as a result of the macro update. This is done by checking the GENASM subentries in the target zone MAC entry. The names in the list can be names of either target zone ASSEM or SRC entries. In either case, SMP/E accesses the appropriate entry and uses the information stored there to perform the required assemblies.

If the SYSMOD replaces or modifies a source (++SRC or ++SRCUPD in SYSMOD), SMP/E uses the distribution zone **SRC entry** to determine the functional and service level of the source and the library containing the source code, and then performs the required assembly.

After an assembly has been done (with data from either the ASSEM entry or the SRC entry), SMP/E knows that the resulting object deck may be link-edited into the distribution libraries somewhere. Although there is no direct connection (that is, no subentry value present) between the ASSEM or SRC entries and a **MOD entry**, there is an implicit relationship. SMP/E assumes that for each ASSEM and SRC entry there exists a MOD entry with the same name. Thus, after performing the assembly, SMP/E accesses the corresponding MOD entry to determine where to install the object deck. This leads us to the same point as if a MOD were supplied in the SYSMOD.

If the SYSMOD replaces or modifies a module (++MOD or ++ZAP in SYSMOD), or if an assembly was done for a SRC or ASSEM entry, SMP/E uses the distribution zone **MOD entry** to determine the functional and service level of the module, the DLIB into which it must be linked, the load modules into which the module should be linked, and the link-edit attributes that should be used. Information about load modules is kept in the LMOD subentries in the MOD entry.

The LMOD subentries within the MOD entry lead SMP/E to the distribution zone **LMOD entries**. These represent load modules that exist in the target libraries. LMOD entries contain all the information necessary, either to relink the load module or to update it using IMASPZAP.

If the SYSMOD replaces a data element (a data element MCS is in the SYSMOD), SMP/E uses the distribution zone **data element entry** to determine the functional and service level of the data element. It then gets the data element installed in the appropriate distribution library.

If the SYSMOD replaces an HFS element (a ++HFS MCS is in the SYSMOD), SMP/E uses the distribution zone **HFS entry** to determine the functional and service level of the HFS element. It then gets the HFS element installed in the appropriate distribution library.

After all the updating, assembling, linking, and so on, are done, you arrive back almost at the starting point, the SYSMOD entry. The difference is that this is the distribution zone SYSMOD entry, which contains all the information about what was done as a result of installing this SYSMOD.

Figure 91 on page 639 shows how the various SMP/E status and structure entries are related. A more detailed explanation of each entry, and the data contained within it, can be found in subsequent sections of this chapter.

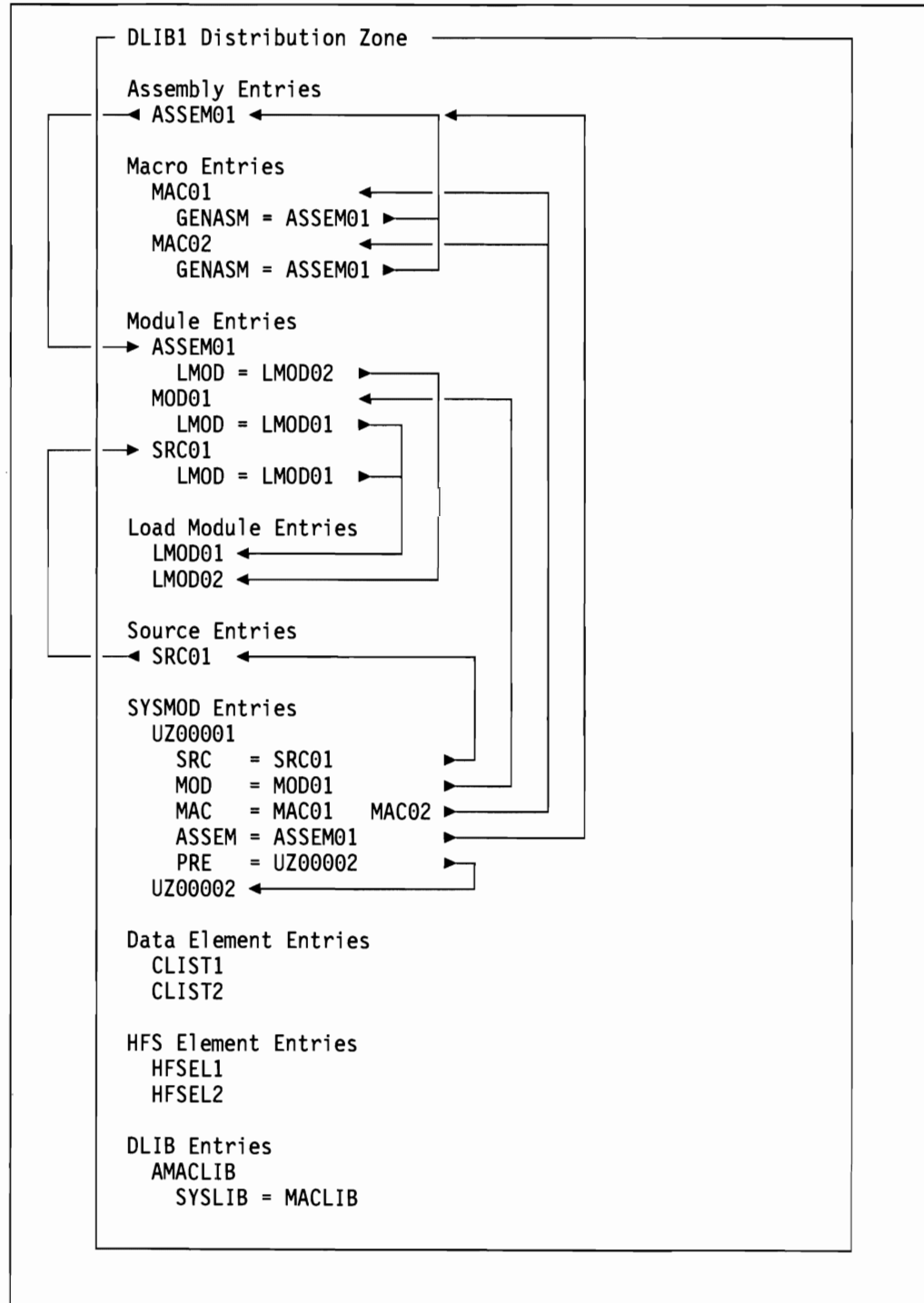


Figure 91. Distribution Zone: Relationships between Entries that Define Status and Structure

**Note:** Lines to the left of the figure are implicit connections based on the assumption that entries with equal names will be found.

The remaining sections of this chapter deal with each of the specific entries in the various SMP/E data sets.

### ASSEM Entry (Distribution and Target Zone)

The ASSEM entry contains assembler statements that can be assembled to create an object module. It is created during JCLIN processing when SMP/E encounters an assembler step with inline assembler input. When the module is reassembled using the statements in the ASSEM entry, SMP/E copies those statements into the SMPWRK2 data set, and then assembles the module.

If a macro is invoked in the assembly, the ASSEM entry is pointed to by the GENASM subentry in the MAC entry created for that macro. As a result, when that macro is updated, SMP/E can reassemble the affected module using the statements in the ASSEM entry. For additional information, see “Processing” on page 179.

### Subentries

These are the subentries for the ASSEM entry as they appear in the LIST output:

*name*

is the name of the ASSEM entry. It corresponds to the name of the module that can be reassembled by use of that ASSEM entry.

The name can contain from 1 to 8 alphanumeric characters.

#### ASSEMBLER INPUT

is the actual assembler statements that were saved for this module during JCLIN processing. These statements are passed to the assembler whenever this module must be reassembled.

The UCL operands are **++ASMIN** and **++ENDASMIN**.

- An ASSEM entry must contain at least the **++ASMIN** and **++ENDASMIN** statements, plus the associated assembler statements.
- The **++ASMIN** and **++ENDASMIN** statements must start in column 1.
- No other operands can start on the same line as the **++ASMIN** statement.
- The **++ENDASMIN** statement must be specified if the **++ASMIN** statement is specified.

#### LASTUPD

identifies the cause of the last change to this ASSEM entry.

The UCL operand is **LASTUPD(value)**. This subentry can contain one of the following values:

#### JCLIN

indicates that the change was made during JCLIN command processing.

#### UCLIN

indicates that the change was made as a result of UCLIN processing.

*sysmod\_id*

indicates that the change was made during the installation of the indicated SYSMOD.

The SYSMOD ID must contain 7 alphanumeric characters.

**LASTUPD TYPE**

indicates how the entry was last changed.

The UCL operand is **LASTUPDTYPE**(*value*). This subentry can contain one of the following values:

- ADD**      The entry was added.
- UPD**      The entry was updated.

**LIST Examples**

To list all the ASSEM entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)            /* Set to requested zone. */.
LIST     ASSEM                /* List all ASSEM entries. */.
```

To list specific ASSEM entries, you can use these commands:

```
SET      BDY(TGT1)            /* Set to requested zone. */.
LIST     ASSEM(ASSEM01        /* List only these two     */
                              /* ASSEM02)                /* entries.                */.
```

The format of the LIST output for each ASSEM entry is the same for both of these commands. The only difference is the number of ASSEM entries listed. Figure 92 is an example of LIST output for ASSEM entries.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
TGT1            ASSEMBLER ENTRIES

NAME
ASSEM01    LASTUPD            = JXY1102 TYPE=ADD
           ASSEMBLER INPUT = ...
                              ... assembler statements
                              ....
ASSEM02    LASTUPD            = JXY1121 TYPE=UPD
           ASSEMBLER INPUT = ...
                              ... assembler statements
                              ....
```

Figure 92. ASSEM Entry: Sample LIST Output

You can use the LIST command to find the MAC entries for macros that are called by the assembler statements in this ASSEM entry. To include the names of these MAC entries in the LIST output, you can use the XREF operand, as shown in these commands:

```
SET      BDY(TGT1)            /* Set to requested zone. */.
LIST     ASSEM                /* List all ASSEM entries */
                              /* XREF                    /* and macros that use them. */.
```

### Notes:

1. XREF can be used either in mass mode or in select mode.
2. SMP/E obtains the data included for the XREF operand by checking the GENASM subentries in all the MAC entries. Because this data is not contained in the ASSEM entry itself, you cannot use UCLIN to change it in the ASSEM entry.

Figure 93 is an example of the LIST output produced when the XREF operand is used.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMLIST OUTPUT
TGT1      ASSEMBLER ENTRIES

NAME
ASSEM01  LASTUPD      = JXY1102 TYPE=ADD
         ASSEMBLER INPUT = ...
         ... assembler statements
         ...
         MACROS USED   = NAME      FMID
                       MAC01     JXY1102
                       MAC02     JXY1121

ASSEM02  LASTUPD      = JXY1121 TYPE=UPD
         ASSEMBLER INPUT = ...
         ... assembler statements
         ...
         MACROS USED   = NAME      FMID
                       MAC01     JXY1102
                       MAC03     JXY1121
```

Figure 93. ASSEM Entry: Sample LIST Output When XREF Is Specified

## UNLOAD Examples

To dump the ASSEM entries in UCL format, you can use the UNLOAD command. To unload all the ASSEM entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested zone.  */.
UNLOAD   ASSEM              /* Unload all ASSEM entries. */.
```

To unload specific ASSEM entries, you can use these commands:

```
SET      BDY(TGT1)          /* Set to requested zone.  */.
UNLOAD   ASSEM(ASSEM01     /* Unload only these two  */.
          ASSEM02)        /* entries.                */.
```

The format of the UNLOAD output for each ASSEM entry is the same for both of these commands. The only difference is the number of ASSEM entries listed. Figure 94 on page 643 is an example of UNLOAD output for ASSEM entries.

```

UCLIN .
REP      ASSEM      ( ASSEM01 )
          LASTUPD   ( JXY1102 )
          LASTUPDTYPE ( ADD )
++ASMIN
...
... assembler statements
...
++ENDASMIN

REP      ASSEM      ( ASSEM02 )
          LASTUPD   ( JXY1121 )
          LASTUPDTYPE ( UPD )
++ASMIN
...
... assembler statements
...
++ENDASMIN

ENDUCL.

```

Figure 94. ASSEM Entry: Sample UNLOAD Output

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the ASSEM entry. When you use UCLIN to update an ASSEM entry, keep these points in mind:

- After the UCLIN changes are done, the ASSEM entry must contain at least ++ASMIN and ++ENDASMIN statements, plus the associated assembler input. Otherwise, there is not enough information in the entry to assemble the associated module.
- The input following the ++ASMIN statement replaces the existing assembler input in the ASSEM entry.
- If a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.
- When SMP/E processes a DEL statement, it does not compare any assembler input after the ++ASMIN statement with the input that is currently in the ASSEM entry. It just deletes the existing assembler input. This causes an error, because there is now insufficient data in the ASSEM entry.

The following examples are provided to help you use the ASSEM entry:

- “Example 1: Deleting an ASSEM Entry”
- “Example 2: Adding a New ASSEM Entry”

### Example 1: Deleting an ASSEM Entry

The main use of UCLIN for an ASSEM entry is to delete the entry. This can be done with the following commands:

```
SET      BDY(TGT1)          /* Set to target zone.    */.  
UCLIN                                /*                        */.  
DEL      ASSEM(ASSEM01)    /* Delete the entry.     */.  
ENDUCL                                /*                        */.
```

### Example 2: Adding a New ASSEM Entry

To create an ASSEM entry, you should run the JCLIN command rather than use UCLIN. (For examples of how JCLIN creates ASSEM entries, see “Examples” on page 173.) However, you can also use UCLIN to create ASSEM entries. For example, you can use the following commands to create a new ASSEM entry and to update an **existing** MAC entry to show that the macro is used in the new assembly:

```
SET      BDY(TGT1)          /* Set to target zone.    */.  
UCLIN                                /*                        */.  
ADD      ASSEM(ASSEM99)    /* New ASSEM entry.     */.  
++ASMIN                                /* Assembler data.     */.  
ASSEM99  CSECT              /*                        */.  
          MAC99  0,0,1      /*                        */.  
          END ASSEM99      /*                        */.  
++ENDASMIN                            /* End of assembler data.*/.  
          /* End of adding ASSEM. */.  
ADD      MAC(MAC99)        /* Modify macro entry used */.  
          GENASM(ASSEM99)  /* to indicate use in new */.  
          /* assembly entry.    */.  
ENDUCL                                /*                        */.
```



## BACKUP Entries (SMPSCDS)

BACKUP entries are a collection of target zone entries that are copied to the SMPSCDS during APPLY processing before they are updated by inline JCLIN, a ++MOVE MCS, or a ++RENAME MCS, or before they are deleted by an element MCS with the DELETE operand. A BACKUP entry is also created for a MOD entry if a SYSMOD being installed provides a ++MOD statement that either changes the distribution library for the module or adds the module to an existing load module.

- As SMP/E processes the inline JCLIN for a given SYSMOD, it determines which entries will be affected by that JCLIN. Before making the changes, it saves a copy of each of those entries on the SMPSCDS.
- Likewise, as SMP/E processes the ++MOVE, ++RENAME, and element MCSs in a given SYSMOD, it determines which entries will be updated or deleted. Before updating or deleting the entries, it saves a copy of each of those entries on the SMPSCDS.

Besides saving copies of the affected entries, SMP/E also saves a SYSMOD entry on the SMPSCDS to indicate which entries will be added or updated by the SYSMOD. Each entry is associated with only one SYSMOD. The entries associated with a SYSMOD are called the BACKUP entries for that SYSMOD. BACKUP entries consist of:

- A SYSMOD entry indicating what entries were added, deleted, or updated
- ASSEM entries for updated target zone ASSEM entries
- LMOD entries for updated target zone LMOD entries
- MAC entries for updated or deleted target zone MAC entries
- MOD entries for updated or deleted target zone MOD entries
- SRC entries for updated or deleted target zone SRC entries
- Data element entries for deleted target zone data element entries
- HFS entries for deleted target zone HFS entries
- DLIB entries for updated target zone DLIB entries

SMP/E provides access to the BACKUP entries as a group—for example, through the LIST command—but it does not provide access to the individual entries or sub-entries.

## Subentries

The subentries in the BACKUP entries are the same as those in the various entry types that are copied. For more information, see the section in this chapter describing each entry.

## LIST Examples

To list the BACKUP entries for all the SYSMODs in the SMPSCDS, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested tgt zone. */.
LIST     BACKUP             /* List all BACKUP entries. */.
```

To list the BACKUP entries for a specific SYSMOD, you can use these commands:

```
SET      BDY(TGT1)          /* Set to requested tgt zone. */.
LIST     BACKUP(UZ12345,    /* List only BACKUP entries */
          UZ12346)          /* for these two SYSMODs. */.
```

The format of the LIST output for each group of BACKUP entries is the same for both of these commands. The only difference is the number of SYSMODs for which BACKUP entries are listed.

When you list the BACKUP entries for a SYSMOD, the first entry in the output is a summary entry for the SYSMOD, which indicates the date and time the SYSMOD was applied, as well as which entries were added or updated as a result of applying the SYSMOD. This is followed by a listing of all the existing target zone entries affected by this SYSMOD, before they were updated. Nothing is listed for added entries, because no entry existed before the SYSMOD was installed.

Figure 95 on page 647 is a partial example of LIST output for BACKUP entries. It shows the summary records, but not all the backup copies of the entries modified by the SYSMOD. This is because the format of those copies is the same as the format for the original target zone entries.

### UCLIN Examples

You can use the DEL UCL statement to delete BACKUP entries from the SMPSCDS. This can be helpful if you plan to do an APPLY followed by ACCEPT when several target libraries have been created from the same distribution library.

When a SYSMOD is accepted into a distribution zone, the entries associated with the SYSMOD are automatically deleted from the SMPSCDS for the RELATED target zone. However, even if the SYSMOD was also applied to other target zones created from the same distribution zone, SMP/E does not clean up the SMPSCDS data sets for the other target zones.

To delete the entries from these data sets, you can accept the SYSMOD and name these other target zones as the RELATED zone. However, this updates the distribution library each time, which is time-consuming and can use up space in the distribution library data set.

Instead, you can use the DEL command to delete these entries without updating the distribution library. To determine which entries to specify, check the SMPLOG data set to see which ones SMP/E deleted during ACCEPT processing.

**Note:** You can also use the CLEANUP command to delete BACKUP entries without specifying them individually. For more information, see Chapter 4.

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
SMPSCDS      BACKUP ENTRIES

NAME
UZ12345  DATE/TIME APP   = 90.100 08:15:00
          ASSEM  (ADD) = ASSEM01 ASSEM02 ASSEM03
          LMOD  (ADD) = LMOD01  LMOD02
          MACRO (ADD) = MAC01
          MOD   (ADD) = MOD01    MOD02    MOD03    MOD04
          SRC   (ADD) = SRC01    SRC02    SRC03
          DLIB  (ADD) = DLIB01
          ASSEM (DEL) = ASSEM04 ASSEM05
          LMOD  (DEL) = LMOD03  LMOD04  LMOD05
          MACRO (DEL) = MAC05    MAC09
          MOD   (DEL) = MOD10    MOD11    MOD12
          SRC   (DEL) = SRC11
          ASSEM (UPD) = ASSEM91 ASSEM92 ASSEM93
          LMOD  (UPD) = LMOD9A  LMOD9B
          MACRO (UPD) = MAC99
          MOD   (UPD) = MOD99
          SRC   (UPD) = SRC99
          DLIB  (UPD) = DLIB99

...
... deleted entries for ASSEM04 ASSEM05
...                      LMOD03 LMOD05 LMOD05
...                      MAC05  MAC09
...                      MOD10  MOD11  MOD12
...                      SRC11
... updated entries for ASSEM91 ASSEM92 ASSEM93
...                      LMOD9A  LMOD9B
...                      MAC99
...                      MOD99
...                      SRC99
...                      DLIB99
... would follow here - format is as in sample
... for each target zone entry
...

```

Figure 95. BACKUP Entries: Sample LIST Output

**Example: Deleting BACKUP Entries**

Assume BACKUP entries exist on the SCDS for SYSMODs UZ12345 and UZ12346. Those SYSMODs have been accepted, but the BACKUP entries have not been deleted from the SCDS. This can happen when multiple systems are supported off one set of DLIBs, where you accept from only one of the target systems. The following UCL could be used to delete the BACKUP entries from the SCDS:

```

SET      BDY(TGT1)          /* Set to TGT1 zone.      */.
UCLIN                               /*                          */.
DEL      BACKUP(UZ12345)    /* Delete the BACKUP entries. */.
DEL      BACKUP(UZ12346)    /* Delete the BACKUP entries. */.
ENDUCL                               /*                          */.

```

**Note:** One UCL statement is required for each BACKUP entry to be deleted.

### Data Element Entry (Distribution and Target Zone)

The data element entry describes an element that is not a macro, module, or source (for example, a CLIST or a sample procedure). Data elements may exist in distribution or target libraries. A data element entry is created the first time you install a SYSMOD that contains an MCS for a data element that does not yet have an entry in the CSI data set.

SMP/E records the function and service level of the data element in the entry. Once a data element entry exists, it is updated as subsequent SYSMODs that affect the data element are installed.

Table 26 on page 523 shows the types of entries used for data elements. Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the entry type contains xxx, which represents the language used for the element. (If an element was not translated, the entry type does not contain any xxx value.) Table 27 on page 526 shows the xxx values and the languages they represent.

### Subentries

These are the subentries for the data element entry as they appear in the LIST output:

#### *name*

is the name of the data element represented by the entry. It can contain from 1 to 8 alphanumeric characters and \$, #, @, or hex C0.

#### **ALIAS**

specifies a list of alias names for the element.

The UCL operand is **ALIAS**(*name...*).

Each alias name can contain from 1 to 8 alphanumeric characters.

#### **DISTLIB**

specifies the ddname of the distribution library for the data element.

The UCL operand is **DISTLIB**(*ddname*).

- The ddname can contain from 1 to 8 alphanumeric characters.
- The DISTLIB subentry is required. Without it, SMP/E cannot process any changes for the data element.

#### **FMID**

specifies the functional owner of this data element. The functional owner is the last function SYSMOD that replaced this element.

The UCL operand is **FMID**(*sysmod\_id*).

The SYSMOD ID must contain 7 alphanumeric characters.

**LASTUPD**

identifies the cause of the last change to this data element entry.

The UCL operand is **LASTUPD**(*value*). This subentry can contain one of the following values:

**UCLIN**

indicates that the change was made as a result of UCLIN processing.

*sysmod\_id*

indicates that the change was made during the installation of the indicated SYSMOD.

The SYSMOD ID must contain 7 alphanumeric characters.

**LASTUPD TYPE**

indicates how the entry was last changed.

The UCL operand is **LASTUPDTYPE**(*value*). This subentry may contain one of the following values:

**ADD**      The entry was added.  
**UPD**      The entry was updated.

**RMID**

identifies the last SYSMOD that **replaced** this data element. Any subsequent SYSMOD that modifies this data element must have a defined relationship (such as PRE or SUP) with this SYSMOD.

The UCL operand is **RMID**(*sysmod\_id*).

- The SYSMOD ID must contain 7 alphanumeric characters.
- If **RMID** is not specified but **FMID** is, SMP/E sets the RMID value to the specified FMID.

**SYSLIB**

specifies the ddname of the target library for the data element.

The UCL operand is **SYSLIB**(*ddname*).

- Only one SYSLIB value may be specified.
- The ddname can contain from 1 to 8 alphanumeric characters.

**LIST Examples**

To list all the data element entries of a given type (such as CLIST) in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */
LIST     CLIST              /* List all CLIST entries. */
```

To list specific CLIST entries, you can use these commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */
LIST     CLIST(CLIST1      /* List only these two
                        CLIST2) /* entries. */
```

The format of the LIST output for each data element entry is the same for both of these commands. The only difference is the number of data element entries listed. Figure 96 is an example of LIST output for data element entries.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMLIST OUTPUT
TGT1          CLIST ENTRIES

NAME
CLIST1  LASTUPD      = MYCLST1 TYPE=ADD
        LIBRARIES   = DISTLIB=AMACLIB  SYSLIB=ISPCLIB
        FMID        = MYCLST1
        RMID        = MYCLST1
CLIST2  LASTUPD      = MYCPTF1 TYPE=ADD
        LIBRARIES   = DISTLIB=AMACLIB  SYSLIB=ISPCLIB
        FMID        = MYCLST1
        RMID        = MYCPTF1
```

Figure 96. Data Element Entry: Sample LIST Output

By specifying the FORFMID operand, you can reduce the number of data element entries listed. When FORFMID is specified, SMP/E lists a data element entry only if its FMID matches one of the FMIDs specified on the FORFMID operand. For example, to list CLIST entries whose FMIDs are defined in FMIDSET TP or else are MYCLST1, you can use these commands:

```
SET      BDY(TGT1)          /* Set to target zone.    */.
LIST     CLIST              /* List all CLIST entries */.
         FORFMID(TP        /* for the TP FMIDSET    */.
         MYCLST1)         /* and FMID MYCLST1.     */.
```

You can use the LIST command to find out the names of all SYSMODs that have modified data elements. To include the names of these SYSMODs in the LIST output, you can use the XREF operand, as shown in these commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
LIST     CLIST              /* List all CLIST entries */.
         XREF               /* and related SYSMODs.  */.
```

### Notes:

1. XREF can be used either in mass mode or in select mode.
2. SMP/E obtains the data included for the XREF operand by checking for entries for this data element in all the SYSMOD entries. Because this data is not contained in the data element entry itself, you cannot use UCLIN to change it in the data element entry.

Figure 97 is an example of the LIST output produced when the XREF operand is used.

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn  DATE mm/dd/yy  TIME hh:mm:ss  GIMSMP LVL 18.1.nn  SMPLIST OUTPUT
TGT1          CLIST ENTRIES

NAME
CLIST1  LASTUPD      = MYCLST1 TYPE=ADD
        LIBRARIES    = DISTLIB=AMACLIB  SYSLIB=ISPCLIB
        FMID         = MYCLST1
        RMID         = MYCLST1
        SYSMOD HISTORY = SYSMOD  TYPE      DATE  MCS  --STATUS--
                        MYCLST1 FUNCTION 90.100 CLIST APP  ACC

CLIST2  LASTUPD      = MYCPTF1 TYPE=ADD
        LIBRARIES    = DISTLIB=AMACLIB  SYSLIB=ISPCLIB
        FMID         = MYCLST1
        RMID         = MYCPTF1
        SYSMOD HISTORY = SYSMOD  TYPE      DATE  MCS  --STATUS--
                        MYCPTF1 PTF      90.150 CLIST APP  ACC
    
```

Figure 97. Data Element Entry: Sample LIST Output When XREF Is Specified

## UNLOAD Examples

To dump the data element entries in UCL format, you can use the UNLOAD command. To unload all the CLIST entries in a particular zone, you can use the following commands:

```

SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD  CLIST              /* Unload all CLIST entries. */.
    
```

To unload specific CLIST entries, you can use these commands:

```

SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD  CLIST(CLIST1
              CLIST2)      /* Unload only these two */.
                          /* entries. */.
    
```

The format of the UNLOAD output for each data element entry is the same for both of these commands. The only difference is the number of data element entries listed. Figure 98 on page 652 is an example of UNLOAD output for CLIST entries.

```
UCLIN .
REP      CLIST      ( CLIST1 )
         LASTUPD    ( MYCLST1 )
         LASTUPDTYPE ( ADD )
         DISTLIB    ( AMACLIB )
         SYSLIB     ( ISPCLIB )
         FMID       ( MYCLST1 )
         RMID       ( MYCLST1 )
         .
REP      CLIST      ( CLIST2 )
         LASTUPD    ( MYCPTF1 )
         LASTUPDTYPE ( ADD )
         DISTLIB    ( AMACLIB )
         SYSLIB     ( ISPCLIB )
         FMID       ( MYCLST1 )
         RMID       ( MYCPTF1 )
         .
ENDUCL.
```

Figure 98. Data Element Entry: Sample UNLOAD Output

By specifying the FORFMID operand, you can reduce the number of data element entries unloaded. When FORFMID is specified, SMP/E unloads a data element entry only if its FMID matches one of the FMIDs specified on the FORFMID operand. For example, to unload CLIST entries whose FMIDs are either defined in FMIDSET TP or are MYCLST1, you can use these commands:

```
SET      BDY(TGT1)      /* Set to target zone.      */.
UNLOAD  CLIST           /* Unload all CLIST entries */
        FORFMID(TP     /* for the TP FMIDSET      */
              MYCLST1) /* and FMID MYCLST1.      */.
```

### UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the data element entry. After the UCLIN changes are made, the data element entry must contain at least the following subentries:

- DISTLIB
- FMID
- RMID

Otherwise, there is not enough information in the entry to process the data element. If any of these subentries are missing, SMP/E does not make the requested UCL updates to the entry, and the entry remains as it was before the UCL command.



**Example: Adding a New Data Element Entry**

Assume you have installed CLIST3 outside of SMP/E, but now you want to start using SMP/E to track changes to that CLIST. Here is an example of the UCL statements you would use to define entries for that CLIST in the appropriate target and distribution zones:

```
SET      BDY(TGT1)          /* Set to target zone.    */.  
UCLIN                               /*                          */.  
ADD      CLIST(CLIST3)     /* Define new CLIST entry.*/  
          DISTLIB(AMACLIB) /* Define DLIB.           */  
          SYSLIB(ISPCLIB)  /* System library.        */  
          FMID(MYCLST1)    /* Functional owner.      */.  
ENDUCL                               /*                          */.  
SET      BDY(DLB1)          /* Now do same to DLIB.   */.  
UCLIN                               /*                          */.  
ADD      CLIST(CLIST3)     /* Define new CLIST entry.*/  
          DISTLIB(AMACLIB) /* Define DLIB.           */  
          /* no SYSLIB info in DLIB. */  
          FMID(MYCLST1)    /* Functional owner.      */.  
ENDUCL                               /*                          */.
```

### DDDEF Entry (Distribution, Target, and Global Zone)

The DDDEF entry contains the information SMP/E needs to dynamically allocate a specific data set. With DDDEF entries, you do not have to provide a DD statement for every data set SMP/E may need to process a particular command. When SMP/E determines that it needs a specific data set, it looks for a DD statement that it can use to allocate that data set. If there is no DD statement, SMP/E checks whether the current zone contains a DDDEF entry for that data set. If so, it uses the information in the DDDEF entry to dynamically allocate the data set.

For more information about dynamically allocating data sets, see Appendix C.

#### Notes:

1. In a job with multiple SET commands, if you use DDDEF entries that specify SYSOUT for SMP/E output (such as SMPOUT or SMPRPT), SMP/E produces multiple SYSOUT data sets. This can cause undesirable results; for example, the output may appear to be out of sequence from one SET command to the next. Therefore, when you run such a job, you may prefer to use DD statements instead of DDDEF entries for SMP/E output data sets.
2. SMP/E does not strictly enforce rules for which subentries can be specified in DDDEF entries for specific data sets. To prevent possible allocation errors, you should refer to the JCL user's guide for your operating system.

### Subentries

These are the subentries for the DDDEF entry as they appear in the LIST output: For more information concerning DDDEF entry syntax, see Chapter 22, "The UCLIN Command" on page 355.

#### *name*

is the ddname of the data set that will be allocated.

- The name can contain from 1 to 8 alphanumeric characters.
- Other than checking for duplicate DDDEF names in a given zone, SMP/E does not check whether the specified name is associated with another data set. For example, SMP/E does not check whether a DDDEF name is the same as a zone name. You must do this checking yourself to avoid undesired results.

#### **BLK(size), CYL, or TRK**

specifies the space units to be used in allocating the data set: blocks, cylinders, or tracks.

The UCL operand is **BLOCK(size)**, **CYLINDERS**, or **TRACKS**.

- *size* is the size, in decimal, of each block to be allocated. The number of blocks to be allocated is specified on SPACE.
- **BLOCK** can also be specified as **BLK**.
- **CYLINDERS** can also be specified as **CYL**.
- **TRACKS** can also be specified as **TRK**.
- These operands are mutually exclusive with each other and with OLD, MOD, SHR, CONCAT, PATH, and SYSOUT.

- You cannot specify these operands for an SMPTLIB DDDEF entry in a distribution zone or a target zone.

### **CATALOG, DELETE, or KEEP**

specifies the final disposition (DISP) of the data set.

The UCL operand is **CATALOG, DELETE, or KEEP**.

- These operands are mutually exclusive with each other and with **CONCAT** and **PATH**.
- You cannot specify these operands for an SMPTLIB DDDEF entry in a distribution zone or a target zone.
- You cannot specify a final disposition for SMPTLIB data sets. SMP/E automatically specifies a final disposition based on the command being processed. For more information, see “SMPTLIB Data Set” on page 618.

### **CONCAT**

identifies one or more DDDEF entries, **existing in the same zone**, that should be concatenated during SMP/E processing.

The UCL operand is **CONCAT**(*name...*).

- The DDDEF names can contain from 1 to 8 alphanumeric characters.
- There must be a DDDEF entry for each specified name. SMP/E does not check other sources (such as DD statements or module GIMMPDFT) to get the information needed to allocate the data sets.
- SMP/E allows you to specify up to 123 names. However, the actual number of partitioned data sets that can be concatenated depends on the operating system you are running under. To determine the maximum number of data sets you can concatenate, see the data management manual for your operating system.
- **CONCAT** is required for concatenated data sets.
- **CONCAT** is mutually exclusive with all other operands.

### **DATACLAS**

specifies the name of a data class to be used for allocating a new data set managed by SMS.

The UCL operand is **DATACLAS**(*name*).

- The data class name is defined by the storage administrator at your installation. This value can contain from 1 to 8 alphanumeric characters (A through Z and 0 through 9) or national characters (@, #, \$) and must start with either an alphabetic or national character.
- This operand can be specified only if your storage is managed by SMS.
- This operand should be specified only when you are allocating a new data set. SMS ignores this parameter if it is specified for an existing data set.
- This operand is mutually exclusive with **PATH** and **CONCAT**.

**Note:** The SMS-related subentries (**DATACLAS**, **MGMTCLAS**, **STORCLAS**, and **DSNTYPE**) **cannot** be specified for a SMPTLIB DDDEF entry in a target or distribution zone.

### DATASET

is the name of the data set to be allocated.

The UCL operand is **DATASET**(*dsname*).

- **DATASET** can also be specified as **DA**.
- The data set name must conform to standard naming conventions for data sets. Each part of the name must contain from 1 to 8 characters, separated from the other parts by a period (.), with no intervening blanks. The maximum length of the entire name is 44 characters (including the periods).
- The data set name itself cannot contain parentheses.
- The data set name can be *NULLFILE* to define a dummy data set.
- **DATASET** is required if **OLD**, **SHR**, or **MOD** is specified.
- **DATASET** is mutually exclusive with **CONCAT**, **PATH**, and **SYSOUT**.
- **DATASET** cannot be used to specify the data set name of an **SMPTLIB** data set. Instead, you can use the **DSPREFIX** operand in either the **SMPTLIB** DDDEF entry or in the **OPTIONS** entry used to process those data sets.
- **SMP/E** does not check whether the specified data set name is unique within the zone. For example, **SMP/E** does not check whether the data set name was also defined in another DDDEF entry in the same zone, or whether the data set name defines the CSI data set containing the zone. You must do this checking yourself to avoid undesired results.

### DIR

specifies the number of directory blocks to allocate.

The UCL operand is **DIR**(*nnnn*).

- The number specified can contain from 1 to 4 decimal digits.
- **DIR** is mutually exclusive with **OLD**, **MOD**, **SHR**, **CONCAT**, and **PATH**.
- You cannot specify **DIR** for an **SMPTLIB** DDDEF entry in a distribution zone or a target zone.

### DSNTYPE

specifies the type of partitioned data set to be created. This operand should be used **only** if you are running **SMP/E** in an **MVS/ESA** environment.

The UCL operand is **DSNTYPE**(**LIBRARY**) or **DSNTYPE**(**PDS**).

#### LIBRARY

specifies that a PDSE (which must be an SMS-managed data set) is to be created.

**Note:** **SMPTLIB** data sets should not be allocated as PDSEs, because **IEBCOPY** does not support copying an unloaded PDS load library from tape to a PDSE load library on **DASD**.

#### PDS

specifies that a PDS is to be created.

**DSNTYPE** can also be specified in a data class, or a member in **SYS1.PARMLIB**.

This operand is mutually exclusive with PATH and CONCAT.

**Note:** The SMS-related subentries (DATACLAS, MGMTCLAS, STORCLAS, and DSNTYPE) **cannot** be specified for a SMPTLIB DDDEF entry in a target or distribution zone.

### DSPREFIX

specifies the data set prefix to be used to construct the full data set name for SMPTLIB data sets. For more information about names for SMPTLIB data sets, see “Receiving SYSMODs Packaged in Relative Files” on page 243.

The UCL operand is **DSPREFIX**(*prefix*).

- The prefix can contain from 1 to 26 alphanumeric characters.
- The prefix must follow standard conventions for naming data sets.
- Instead of specifying **DSPREFIX** in the SMPTLIB DDDEF entry, you can specify it in the OPTIONS entry that is in effect when you receive RELFILEs into the SMPTLIB data sets.

If you do not specify a data set prefix in the SMPTLIB DDDEF entry or in the appropriate OPTIONS entry, no prefix is included when SMP/E assigns a name to the SMPTLIB data sets.

- DSPREFIX is mutually exclusive with CONCAT and PATH.

**Note:** This subentry exists only in the global zone.

### MGMTCLAS

specifies the name of a management class to be used for allocating a new data set managed by SMS.

The UCL operand is **MGMTCLAS**(*name*).

- The management class name is defined by the storage administrator at your installation. This value can contain from 1 to 8 alphanumeric characters (A through Z and 0 through 9) or national characters (@, #, \$) and must start with either an alphabetic or national character.
- This operand can be specified only if your storage is managed by SMS.
- This operand should be specified only when you are allocating a new data set. SMS ignores this parameter if it is specified for an existing data set.
- This operand is mutually exclusive with PATH and CONCAT.

**Note:** The SMS-related subentries (DATACLAS, MGMTCLAS, STORCLAS, and DSNTYPE) **cannot** be specified for a SMPTLIB DDDEF entry in a target or distribution zone.

### MOD, NEW, OLD, or SHR

specifies the initial disposition (DISP) of the data set.

The UCL operand is **MOD**, **NEW**, **OLD**, or **SHR**.

- One of these operands must be specified if the data set is not cataloged. If one of these operands is specified for a cataloged data set, it overrides the value in the catalog.
- These operands are mutually exclusive with each other and with CONCAT and PATH.

OLD, SHR, and MOD are also mutually exclusive with SYSOUT, PROTECT, BLOCK, CYLINDERS, TRACKS, DIR, and SPACE.

- You cannot specify an initial disposition for SMPTLIB data sets. SMP/E automatically specifies an initial disposition, which is based on the command being processed. For more information, see “SMPTLIB Data Set” on page 618.

### PATH

identifies the name of the path to be allocated in the hierarchical file system. The name is not a complete pathname; it is a directory. This value is concatenated with the appropriate element or load module name to create a complete pathname.

The UCL operand is **PATH**(*pathname*).

- PATH is mutually exclusive with all other DDDEF entry operands.
- PATH is meaningful only in a target zone DDDEF entry, because the information is used only for processing a target zone.
- The pathname can be from 1 to 255 characters.
- The pathname must begin and end with a slash (/).
- In addition to the required delimiters (/), a pathname must also be enclosed in single apostrophes (') if any of the following is true:
  - The pathname contains lowercase alphabetic characters.
  - The pathname contains a character that is not uppercase alphabetic, numeric, national (\$, #, or @), slash (/), plus (+), hyphen, period, or ampersand (&).
  - The pathname spans more than one line in the UCL control statement.

The apostrophes must be outside the required delimiters, as in '/pathname/', not '/pathname'/.

The single apostrophes used to enclose a pathname (the delimiters) do not count as part of the 255-character limit.

- Any apostrophes specified as part of a pathname (not the delimiters) must be doubled.

Double apostrophes count as two characters in the 255-character limit.

- The pathname can include characters X'40' through X'FE'.
- Because symbolic substitution is not detected by SMP/E, it is not supported by SMP/E.

### PROTECT

specifies that the RACF PROTECT option should be used when a new data set is first allocated. If PROTECT is specified and RACF is installed, the data set allocated by SMP/E will be RACF-protected.

PROTECT can also be used to indicate that an existing data set is RACF-protected. In this case, because SMP/E does not allocate the data set, it does not check the PROTECT indicator. However, you can use it to keep a record of which data sets have been protected with RACF.

The UCL operand is **PROTECT**.

- PROTECT is mutually exclusive with CONCAT, PATH, OLD, MOD, and SHR.
- You cannot specify PROTECT for an SMPTLIB DDDEF entry in a distribution zone or a target zone.

### SPACE

specifies the primary and secondary space allocation for new data sets.

The UCL operand is **SPACE**(*prime,second*).

- Each value must contain from 1 to 4 decimal digits, and the two values must be separated by a comma or a blank.
- SPACE is mutually exclusive with CONCAT, PATH, OLD, MOD, and SHR.

### STORCLAS

specifies the name of a storage class used for allocating a new data set managed by SMS.

The UCL operand is **STORCLAS**(*name*).

- The storage class name is defined by the storage administrator at your installation. This value can contain from 1 to 8 alphanumeric characters (A through Z and 0 through 9) or national characters (@, #, \$) and must start with either an alphabetic or national character.
- This operand can be specified only if your storage is managed by SMS.
- This operand should be specified only when you are allocating a new data set. SMS ignores this parameter if it is specified for an existing data set.
- This operand is mutually exclusive with PATH and CONCAT.

**Note:** The SMS-related subentries (DATACLAS, MGMTCLAS, STORCLAS, and DSNTYPE) **cannot** be specified for a SMPTLIB DDDEF entry in a target or distribution zone.

### SYSOUT

specifies the output class for SYSOUT data sets.

The UCL operand is **SYSOUT**(*value*).

- If a class is specified, it must be 1 alphabetic or numeric character (A through Z or 0 through 9).
- If \* is specified, the output class depends on references to an OUTPUT JCL statement.
  - If there is an implicit or explicit reference to an OUTPUT JCL statement, the output is written to the same class as the CLASS parameter on the OUTPUT statement.
  - If there is no reference to an OUTPUT JCL statement, the output is written to the same class as the one specified as MSGCLASS on the JOB card.
- SYSOUT is mutually exclusive with BLOCK, CYLINDER, TRACK, CONCAT, PATH, and DATASET.
- SYSOUT cannot be specified for SMPTLIB data sets.

### UNIT

specifies the UNIT type the data set resides on if it is not cataloged.

The UCL operand is **UNIT**(*type*).

- **UNIT** must be specified if the data set is not cataloged (unless it is not cataloged because of SMS). If it is specified for a cataloged data set, it overrides the value in the catalog.
- The UNIT value can contain from 1 to 8 characters and should conform to standard UNIT naming conventions.  
SMP/E accepts any nonblank characters specified between the open and close parentheses, up to a maximum length of 8.
- UNIT is mutually exclusive with CONCAT and PATH.

### VOLUME

specifies the volume serial number of the volume that the data set resides on if not cataloged.

The UCL operand is **VOLUME**(*valid...*).

- **VOLUME** must be specified if the data set is not cataloged (unless it is not cataloged because of SMS). If **VOLUME** is specified for a cataloged data set, it overrides the value in the catalog.
- The volume identifier can contain from 1 to 6 alphanumeric characters.
- For SMPTLIB data sets, up to five volume serial numbers can be specified. All the volumes must have the same UNIT type. For other data sets, only one volume serial number can be specified.
- VOLUME is mutually exclusive with CONCAT and PATH.

### WAIT=YES or WAIT= NO

indicates whether SMP/E should wait for the data set to be allocated if the volume is not mounted or if the data set is already in use. Not waiting causes allocation to fail for the data set.

The UCL operand is **WAITFORDSN**.

- **WAITFORDSN** can also be specified as **WAIT**.
- WAIT is mutually exclusive with CONCAT and PATH.
- If no value is specified, the default is not to wait.
- WAIT is not related to the PROCESS parameter specified on the EXEC statement. PROCESS affects how long a job should wait for a data set before being run. For more information, see Appendix B, "Sample SMP/E Cataloged Procedure."

## LIST Examples

To list all the DDDEF entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)      /* Set to requested target. */.  
LIST     DDDEF          /* List all DDDEF entries. */.
```



To list specific DDDEF entries in a particular zone, you can use these commands:

```

SET      BDY(TGT1)          /* Set to requested target. */.
LIST     DDDEF(SMPMTS      /* List only these three    */.
          MACLIB           /* entries.                  */.
          SYSLIB)         /*                          */.
    
```

The format of the LIST output for each DDDEF entry is the same for both of these commands. The only difference is the number of DDDEF entries listed.

Figure 99 and Figure 100 on page 662 are examples of LIST output for DDDEF entries.

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
TGT1      DDDEF ENTRIES

NAME

AGENLIB   DATASET      = SYS1.AGENLIB
          VOLUME       = DLIB02
          UNIT         = 3380
          SHR

AMACLIB   DATASET      = SYS1.AMACLIB
          VOLUME       = DLIB01
          UNIT         = 3380
          SHR

BPXLIB1   PATH         = '/path_name1/'

CMACLIB   DATASET      = SYS1.MACLIB
          SHR

MACLIB    DATASET      = SYS1.MACLIB
          OLD

SMPMTS    DATASET      = SYS1.SMPMTS
          OLD

SMPLOG    DATASET      = SYS1.SMPLOG
          MOD

SYSLIB    CONCAT      = SMPMTS   CMACLIB   AMACLIB   AGENLIB

SMPOUT    SYSOUT      = A

SMPWRK1   UNIT        = SYSDA
          SPACE       = (25,25)
          DIR         = 25
          ALLOC       = TRK
          NEW
          DELETE

SMPWRK2   DATACLAS   = FB80CLAS
          MGMTCLAS    = SMPMCLAS
          STORCLAS    = SMPESCLS

SMPTLIB   VOLUME      = DLIB01   DLIB02   DLIB03
          UNIT        = 3380
          PROTECT
    
```

Figure 99. DDDEF Entry: Sample LIST Output for a Target Zone

## DDDEF Entry (Distribution, Target, and Global Zone)

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
GLOBAL      DDDEF ENTRIES

NAME

SMPPTS     DATASET      = SYS1.SMPPTS
           OLD

SMPLOG     DATASET      = SYS1.GLOBAL.SMPLOG
           MOD

SMPOUT     SYSOUT       = A

SMPWRK1    UNIT         = SYSDA
           SPACE       = (25,25)
           DIR         = 25
           ALLOC       = TRK
           NEW
           DELETE

SMPWRK2    DATACLAS   = FB80CLAS
           MGMTCLAS   = SMPMCLAS
           STORCLAS   = SMPESCLS

SMPTLIB    VOLUME      = DLIB01  DLIB02  DLIB03
           UNIT       = 3380
           DSPREFIX   = C87MVSP
           PROTECT
```

Figure 100. DDDEF Entry: Sample LIST Output for a Global Zone

## UNLOAD Examples

To dump the DDDEF entries in UCL format, you can use the UNLOAD command. To unload all the DDDEF entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested target. */.
UNLOAD  DDDEF              /* Unload all DDDEF entries. */.
```

To unload specific DDDEF entries in a particular zone, you can use these commands:

```
SET      BDY(TGT1)          /* Set to requested target. */.
UNLOAD  DDDEF(SMPMTS      /* Unload only these three */
           MACLIB         /* entries. */.
           SYSLIB)       /* */.
```

The format of the UNLOAD output for each DDDEF entry is the same for both of these commands. The only difference is the number of DDDEF entries unloaded.

**Note:** You can use the UNLOAD command only for target and distribution zones, not for the global zone.

Figure 101 is an example of UNLOAD output for DDDEF entries.

```

UCLIN
REP      DDDEF      ( AGENLIB )
        DATASET    ( SYS1.AGENLIB )
        VOLUME     ( DLIB02 )
        UNIT       ( 3380 )
        SHR
        .
REP      DDDEF      ( AMACLIB )
        DATASET    ( SYS1.AMACLIB )
        VOLUME     ( DLIB01 )
        UNIT       ( 3380 )
        SHR
        .
REP      DDDEF      ( BPXLIB1 )
        PATH       ( '/path_name1/' )
        .
REP      DDDEF      ( CMACLIB )
        DATASET    ( SYS1.MACLIB )
        SHR
        .
REP      DDDEF      ( MACLIB )
        DATASET    ( SYS1.MACLIB )
        OLD
        .
REP      DDDEF      ( SMPMTS )
        DATASET    ( SYS1.SMPMTS )
        OLD
        .
REP      DDDEF      ( SMPLOG )
        DATASET    ( SYS1.SMPLOG )
        MOD
        .
REP      DDDEF      ( SYSLIB )
        CONCAT     ( SMPMTS CMACLIB AMACLIB AGENLIB )
        .
REP      DDDEF      ( SMPMLIB )
        VOLUME     ( DLIB01 DLIB02 DLIB03 )
        UNIT       ( 3380 )
        PROTECT
ENDUCL.

```

Figure 101. DDDEF Entry: Sample UNLOAD Output

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in a DDDEF entry. When you use UCLIN to update a DDDEF entry, keep these points in mind:

- If a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.
- When SMP/E dynamically allocates a concatenation list, the order of the concatenation is the same as that specified in the DDDEF entry. Therefore, you cannot use ADD statements to update a DDDEF entry that already contains a concatenation list. SMP/E would not know the correct position for the new data.

For example, suppose you have a DDDEF entry for the SYSLIB data set that concatenates SMPPTS and MACLIB. You **cannot** use the following commands to add XYZMACS at the end of the list:

```
SET      BDY(DLIB1)          /* Set to DLIB1 zone.    */.  
UCLIN                               /*                        */.  
ADD      DDDEF(SYSLIB)      /*                        */.  
          CONCAT(XYZMACS)   /*                        */.  
ENDUCL                               /*                        */.
```

Instead, you can use these commands to replace the entire concatenation list:

```
SET      BDY(DLIB1)          /* Set to DLIB1 zone.    */.  
UCLIN                               /*                        */.  
REP      DDDEF(SYSLIB)      /*                        */.  
          CONCAT(SMPPTS,    /* Replace entire list.  */  
                MACLIB,     /*                        */  
                XYZMACS)    /*                        */.  
ENDUCL                               /*                        */.
```

The following examples are provided to help you use the DDDEF entry:

- “Example 1: Defining Global Zone DDDEFs for Cataloged Data Sets”
- “Example 2: Defining DLIB Zone DDDEFs for Cataloged Data Sets” on page 665
- “Example 3: Defining Target Zone DDDEFs for Cataloged Data Sets” on page 665
- “Example 4: Defining a DDDEF for a Noncataloged Data Set” on page 666
- “Example 5: Defining a Concatenated DDDEF Entry” on page 666
- “Example 6: Defining DDDEFs for Temporary Data Sets” on page 667
- “Example 7: Defining a Global Zone DDDEF for SMPTLIB Data Sets” on page 668
- “Example 8: Defining a DLIB Zone DDDEF for SMPTLIB Data Sets” on page 668
- “Example 9: Protecting Data Sets” on page 668
- “Example 10: Defining Pathnames in a Hierarchical File System” on page 669

### Example 1: Defining Global Zone DDDEFs for Cataloged Data Sets

For this example, assume you want to define a DDDEF entry for the SMPPTS and SMPLOG. The data set names are SYS1.SMPPTS and SYS1.GLOBAL.SMPLOG, and both data sets are cataloged on the system that you are running on. The DDDEF entries should be set up as follows:

```
SET      BDY(GLOBAL)        /* Set to global zone.   */.  
UCLIN                               /*                        */.  
ADD      DDDEF(SMPPTS)      /* DDDEF for PTS.       */.  
          DA(SYS1.SMPPTS)   /* Data set is cataloged.*/  
          OLD               /* DISP=OLD.            */.  
ADD      DDDEF(SMPLOG )    /* DDDEF for LOG.       */.  
          DA(SYS1.SMPLOG)   /* Data set is cataloged.*/  
          MOD               /* DISP=MOD.            */.  
ENDUCL                               /*                        */.
```

**Example 2: Defining DLIB Zone DDDEFs for Cataloged Data Sets**

Assume you want SMP/E to dynamically allocate distribution libraries AOS12 and ASAMPLIB during ACCEPT. Assuming these libraries are cataloged on the system you are running SMP/E on, you can use the following commands to define the DDDEF entries:

```

SET      BDY(DLIB1)          /* Set to DLIB zone.      */
UCLIN                               /*                          */
ADD      DDDEF(AOS12)        /* DLIB ddname.           */
          DA(SYS1.AOS12)     /* Assume cataloged.      */
          OLD                 /* OLD for update.        */
ADD      DDDEF(ASAMPLIB)     /* DLIB ddname.           */
          DA(SYS1.ASAMPLIB)  /* Assume cataloged.      */
          OLD                 /* OLD for update.        */
ENDUCL                               /*                          */

```

**Note:** Both data sets have a disposition of OLD. This is because these data sets are used for output when SMP/E calls the system utilities, and you want only one job to update the data sets at a time.

**Example 3: Defining Target Zone DDDEFs for Cataloged Data Sets**

Assume you want SMP/E to dynamically allocate data sets MODGEN, AGENLIB, MACLIB, SMPMTS, and SMPLOG during APPLY. Assuming these data sets are cataloged on the system that you are running SMP/E on, you can use the following commands to define the DDDEF entries:

```

SET      BDY(TGT1)          /* Set to target zone.    */
UCLIN                               /*                          */
ADD      DDDEF(MODGEN )     /* For concatenation.     */
          DA(SYS1.MODGEN)   /* Data set is cataloged.*/
          SHR               /* SHR for read.          */
ADD      DDDEF(AGENLIB )   /* For concatenation.     */
          DA(SYS1.AGENLIB)  /* Data set is cataloged.*/
          SHR               /* SHR for read.          */
ADD      DDDEF(MACLIB )    /* In case updated.       */
          DA(SYS1.MACLIB)   /* Data set is cataloged.*/
          OLD               /* OLD during update.     */
ADD      DDDEF(SMPMTS )    /* For update.            */
          DA(SYS1.SMPMTS)   /* Data set is cataloged.*/
          OLD               /* OLD for update.        */
ADD      DDDEF(SMPLOG )    /* For update.            */
          DA(SYS1.SMPLOG)   /* Data set is cataloged.*/
          MOD               /* MOD for log.           */
ENDUCL                               /*                          */

```

**Note:** MACLIB and SMPMTS have a disposition of OLD. This is because these data sets are used for output when SMP/E calls the system utilities, and you want only one job to update the data sets at a time.

**Example 4: Defining a DDDEF for a Noncataloged Data Set**

During APPLY processing, SMP/E may need to refer to some distribution library data sets not cataloged on the system that SMP/E is running on. For example, suppose AMODGEN is a distribution macro library not cataloged on the running system, and you want to use AMODGEN in the SYSLIB concatenation for assemblies. You can use the following commands to define the DDDEF entry:

```

SET      BDY(TGT1)           /* Set to target zone.      */
UCLIN                               /*                          */
ADD      DDDEF(AMODGEN )     /* For concatenation.       */
          DA(SYS1.AMODGEN)   /* Data set not cataloged. */
          VOLUME(DLIB01)    /*                          */
          UNIT(3330-1)       /*                          */
          SHR                 /* SHR for read.           */
ENDUCL                               /*                          */

```

**Note:** This data set has a disposition of SHR. This is because it is not updated during APPLY, but rather is used just for input.

**Example 5: Defining a Concatenated DDDEF Entry**

After you have defined DDDEF entries for all the data sets to be included in the SYSLIB concatenation, you can define the DDDEF entry that concatenates the data sets. This entry must specify the concatenation ddname and the order of concatenation. The ddname is the name of the DDDEF entry, and the order of concatenation is the order in which the DDDEF names are specified on the CONCAT operand.

**Note:** If you are using MVS/XA or MVS/ESA, the correct sequence is to place MACLIB before MODGEN. Placing MODGEN before MACLIB is appropriate only for MVS/370 users.

You can use the following commands to define the DDDEF entry:

```

SET      BDY(TGT1)           /* Set to target zone.      */
UCLIN                               /*                          */
ADD      DDDEF(SYSLIB)       /* ddname for concatenation.*/
          CONCAT(            /* Concatenation order is  */
            SMPMTS           /* SMPMTS first,           */
            MACLIB           /* then MACLIB,            */
            MODGEN           /* then MODGEN,            */
            AMODGEN          /* then AMODGEN,           */
            AGENLIB          /* then AGENLIB.           */
          )                  /* End of list.            */
ENDUCL                               /*                          */

```

The SYSLIB concatenation is the same as if the following JCL were used:

```

//SYSLIB DD DSN=SYS1.SMPMTS,DISP=OLD
//        DD DSN=SYS1.MACLIB,DISP=OLD
//        DD DSN=SYS1.MODGEN,DISP=OLD
//        DD DSN=SYS1.AMODGEN,DISP=SHR,
//          UNIT=3330-1,VOL=SER=DLIB01
//        DD DSN=SYS1.AGENLIB,DISP=SHR

```

**Note:** SYS1.MACLIB was allocated as DISP=OLD, even though it is not updated during APPLY processing. This is to limit access by any other job while SMP/E is running. If you wanted to use a disposition of SHR when MACLIB is part of the concatenation, you can use the following commands to define

an additional DDDEF entry for MACLIB and change the SYSLIB DDDEF entry:

```

UCLIN                /*                */
ADD      DDDEF(CMACLIB) /* For read access.    */
          DA(SYS1.MACLIB) /* Data set is cataloged. */
          SHR          /* Change DISP to SHR.   */
ADD      DDDEF(SYSLIB)  /* ddname for concatenation. */
          CONCAT(      /* concatenation order is */
                    SMPMTS /* SMPMTS first,         */
                    CMACLIB /* ***** NOTE CHANGE ***** */
                    MODGEN /* then MODGEN,          */
                    AMODGEN /* then AMODGEN,         */
                    AGENLIB /* then AGENLIB.         */
          )           /* End of list.         */
ENDUCL              /*                */

```

This corresponds to the following JCL:

```

//SYSLIB DD DSN=SYS1.SMPMTS,DISP=OLD
//      DD DSN=SYS1.CMACLIB,DISP=SHR
//      DD DSN=SYS1.MODGEN,DISP=OLD
//      DD DSN=SYS1.AMODGEN,DISP=SHR,
//          UNIT=3330-1,VOL=SER=DLIB01
//      DD DSN=SYS1.AGENLIB,DISP=SHR

```

SYS1.MACLIB is now allocated with DISP=SHR so other jobs can still access it.

### Example 6: Defining DDDEFs for Temporary Data Sets

Assume you want SMP/E to dynamically allocate the SMPDOUT and SMPWRK1 data sets during ACCEPT. You can use the following commands to define the DDDEF entries:

```

SET      BDY(DLIB1)    /* Set to DLIB zone.    */
UCLIN    /*                */
ADD      DDDEF(SMPDOUT) /* SMPDOUT ddname.     */
          SYSOUT(A)    /* SYSOUT class.       */
ADD      DDDEF(SMPWRK1) /* SMPWRK1 ddname.    */
          NEW          /* New data set.       */
          DELETE      /* Delete when finished. */
          TRACKS      /* Allocate in tracks.  */
          SPACE(25,25) /* prime and secondary space. */
          DIR(25)     /* 25 directory blocks. */
          UNIT(SYSDA) /* Allocate on SYSDA.  */
ADD      DDDEF(SMPWRK2) /* SMPWRK2 ddname.     */
          DATACLAS(FB80CLAS) /* Add DATACLAS name. */
          MGMTCLAS(SMPMCLS) /* Add MGMTCLAS name.  */
          STORCLAS(SMPESCLS) /* Add STORCLAS name.  */
ENDUCL    /*                */

```

### Example 7: Defining a Global Zone DDDEF for SMPTLIB Data Sets

Assume you want SMP/E to dynamically allocate the SMPTLIB data sets when it receives SYSMODs packaged in relative files. In addition, you want these data sets to be RACF-protected. You can use the following commands to define the DDDEF entry:

```

SET      BDY(GLOBAL)      /* Set to global zone.      */
UCLIN   /*                  */
ADD     DDDEF(SMPTLIB)    /* SMPTLIB ddname.         */
        DATACLAS(FB80CLAS) /* Add DATACLAS name.     */
        DSPREFIX(C87MVSP)  /* Data set prefix.        */
        MGMTCLAS(SMPMCLS)  /* Add MGMTCLAS name.      */
        STORCLAS(SMPESCLS) /* Add STORCLAS name.     */
        PROTECT           /* Request RACF protection. */
ENDUCL  /*                  */
    
```

**Note:** No initial or final disposition is specified. This is because when SMP/E dynamically allocates SMPTLIB data sets, it automatically specifies the initial and final disposition based on the command it is processing.

### Example 8: Defining a DLIB Zone DDDEF for SMPTLIB Data Sets

Assume you want SMP/E to dynamically allocate the SMPTLIB data sets when it accepts SYSMODs packaged in relative files. You can use the following commands to define the DDDEF entry:

```

SET      BDY(DLIB1)      /* Set to DLIB zone.       */
UCLIN   /*                  */
ADD     DDDEF(SMPTLIB)    /* SMPTLIB ddname.         */
        VOLUME(DLIB01)   /* Volumes used - DLIB01  */
        DLIB02           /* DLIB02                  */
        DLIB03           /* DLIB03                  */
        UNIT(3380)       /* Allocate on 3380.      */
ENDUCL  /*                  */
    
```

**Note:** No initial or final disposition is specified. This is because when SMP/E dynamically allocates SMPTLIB data sets, it automatically specifies the initial and final disposition based on the command it is processing.

### Example 9: Protecting Data Sets

Assume you have previously defined your SMPMTS and SMPSTS data sets outside of SMP/E and protected them with RACF. You now want to have these dynamically allocated, and you also want to keep a record that they are RACF-protected. You can use the following commands to define the DDDEF entries:

```

SET      BDY(TGT1)      /* Set to target zone.     */
UCLIN   /*                  */
ADD     DDDEF(SMPMTS)    /* SMPMTS ddname.         */
        DA(SYS1.SMPMTS)  /* Data set is cataloged.  */
        OLD              /* OLD for update.        */
        PROTECT         /* Is RACF-protected.     */
ADD     DDDEF(SMPSTS)    /* SMPSTS ddname.         */
        DA(SYS1.SMPSTS)  /* Data set is cataloged.  */
        OLD              /* OLD for update.        */
        PROTECT         /* Is RACF-protected.     */
ENDUCL  /*                  */
    
```



**Example 10: Defining Pathnames in a Hierarchical File System**

Assume a product you plan to add to your system contains load modules and elements that will reside in a hierarchical file system (HFS). You want to have the pathnames dynamically allocated instead of having DD statements for them in a cataloged procedure. You can use the following commands to define the DDDEF entry for each pathname:

```

-----+-----1-----+-----2-----    ...    -----5-----+-----6-----+-----7--
SET      BDY(TGT1)                /* Set to target zone.      */.
UCLIN                                         /*                          */.
ADD      DDDEF(BPXLIB1)           /* Specify DDDEF name.     */.
        PATH('/path_name1/')      /* Define pathname.        */.
                                         /*                          */.
ADD      DDDEF(BPXLIB2)           /* Specify DDDEF name.     */.
        PATH('/path_name2/This/pathname/is/an/example/of/a/very/long/pat
hname/It/shows/a/long/name/')    /* Define pathname.        */.
                                         /*                          */.
ENDUCL                                         /*                          */.

```

### DLIB Entry (Distribution and Target Zone)

The DLIB entry describes a distribution library that was totally copied to a target library. It is created during JCLIN processing when SMP/E encounters a COPY statement without any SELECT statements. The INDD value on the COPY statement becomes the name of the DLIB entry, and the OUTDD value on the COPY statement becomes the SYSLIB value in the DLIB entry. For additional information, see “Processing” on page 179.

SMP/E uses the DLIB entry to determine where elements and load modules should be applied, as follows:

- **Element entries other than MOD**

- If the element entry has a SYSLIB value, SMP/E checks whether the DISTLIB value in the element entry matches the name of a DLIB entry. If there is a match, and if the SYSLIB value in the DLIB entry is different from the one in the element entry, SMP/E installs the element in the library indicated by the SYSLIB value in the DLIB entry, as well as to the library indicated by the element's SYSLIB value.
- If the element entry has no SYSLIB value, SMP/E looks for a DLIB entry whose name matches the DISTLIB value for the element. SMP/E applies the element to the library indicated by the SYSLIB value in the DLIB entry, and adds that SYSLIB value to the element entry.

- **MOD entries**

For MOD entries, processing is slightly different, because MOD entries do not contain a SYSLIB subentry.

- For a MOD entry that points to an LMOD entry, SMP/E checks whether the DISTLIB value in the MOD entry matches the name of a DLIB entry. If there is a match, and if the SYSLIB value in the DLIB entry is different from the ones in the LMOD entry, the SYSLIB value in the DLIB entry is used in addition to the LMOD's SYSLIB values.
- For a MOD entry that does not point to an LMOD entry, SMP/E checks whether the DISTLIB value in the MOD entry matches the name of a DLIB entry. If there is a match, SMP/E assumes a load module with the same name as the MOD entry should exist in the library specified by the SYSLIB value in the DLIB entry.
  - If there is already an LMOD entry with the same name as the MOD entry, and that LMOD entry does not already contain two SYSLIB values, SMP/E adds the SYSLIB value from the DLIB entry to the LMOD entry.
  - If there is no LMOD entry with the same name as the MOD entry, SMP/E creates one and adds the SYSLIB value from the DLIB entry to the LMOD entry.

Lastly, SMP/E updates the MOD entry by adding an LMOD subentry for the LMOD entry that was updated or created.

## Subentries

These are the subentries for the DLIB entry as they appear in the LIST output:

### *name*

is the name of the distribution library represented by the DLIB entry.

The name can contain from 1 to 8 alphanumeric characters.

### **LASTUPD**

identifies the cause of the last change to this DLIB entry.

The UCL operand is **LASTUPD**(*value*). This subentry can contain one of the following values:

#### **JCLIN**

indicates that the change was made during JCLIN command processing.

#### **UCLIN**

indicates that the change was made as a result of UCLIN processing.

#### *sysmod\_id*

indicates that the change was made during the installation of the indicated SYSMOD.

The SYSMOD ID must contain 7 alphanumeric characters.

### **LASTUPD TYPE**

indicates how the entry was last changed.

The UCL operand is **LASTUPDTYPE**(*value*). This subentry can contain one of the following values:

**ADD**      The entry has been added.  
**UPD**      The entry has been updated.

### **SYSTEM LIBRARIES**

identifies the ddnames of the target libraries into which the distribution library should be copied.

The UCL operand is **SYSLIB**(*ddname...*).

- A DLIB entry used for modules can contain one or two SYSLIB values.
- A DLIB entry used for macros, source, or data elements should contain only one SYSLIB value.
- The ddnames can contain from 1 to 8 alphanumeric characters.

## LIST Examples

To list all the DLIB entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
LIST     DLIB              /* List all DLIB entries. */.
```

## DLIB Entry (Distribution and Target Zone)

To list specific DLIB entries, you can use these commands:

```
SET      BDY(TGT1)           /* Set to requested zone. */.
LIST     DLIB(AMACLIB        /* List only these two    */.
          MACDLB01)          /* entries.                */.
```

The format of the LIST output for each DLIB entry is the same for both of these commands. The only difference is the number of DLIB entries listed. Figure 102 is an example of LIST output for DLIB entries.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
TGT1      DLIB ENTRIES

NAME
AMACLIB   SYSTEM LIBRARY = MACLIB
MACDLB01  SYSTEM LIBRARY = MACLIB   MACTGT01
MODDLB01  LASTUPD        = JXY1102 TYPE=ADD
          SYSTEM LIBRARY = LPALIB   MODTGT01
MODDLB02  LASTUPD        = JXY1121 TYPE=UPD
          SYSTEM LIBRARY = LPALIB   MODTGT02
```

Figure 102. DLIB Entry: Sample LIST Output

## UNLOAD Examples

To dump the DLIB entries in UCL format, you can use the UNLOAD command. To unload all the DLIB entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)           /* Set to requested zone. */.
UNLOAD   DLIB                 /* Unload all DLIB entries. */.
```

To unload specific DLIB entries, you can use these commands:

```
SET      BDY(TGT1)           /* Set to requested zone. */.
UNLOAD   DLIB(AMACLIB        /* Unload only these two  */.
          MACDLB01)          /* entries.                */.
```

The format of the UNLOAD output for each DLIB entry is the same for both of these commands. The only difference is the number of DLIB entries listed. Figure 103 on page 673 is an example of UNLOAD output for DLIB entries.

```

UCLIN .
REP    DLIB      ( AMACLIB )
       SYSLIB    ( MACLIB   )
       .
REP    DLIB      ( MACDLB01 )
       SYSLIB    ( MACLIB   MACTGT01 )
       .
REP    DLIB      ( MODDLB01 )
       LASTUPD   ( JXY1102 )
       LASTUPDTYPE ( ADD   )
       SYSLIB    ( LPALIB   MODTGT01 )
       .
REP    DLIB      ( MODDLB02 )
       LASTUPD   ( JXY1121 )
       LASTUPDTYPE ( UPD   )
       SYSLIB    ( LPALIB   MODTGT02 )
       .
ENDUCL.

```

Figure 103. DLIB Entry: Sample UNLOAD Output

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the DLIB entry. When you use UCLIN to update a DLIB entry, keep these points in mind:

- After the UCLIN changes are made, the DLIB entry must contain at least a SYSLIB subentry. Otherwise, there is not enough information in the entry to indicate where elements should be installed.
- If a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.

### Example: Changing the Destination of a Copied Library

Assume you are doing a system generation and have completed the following steps:

1. Performed stage 1 system generation
2. Allocated new target zone
3. Copied distribution zone to new target zone
4. Ran SMP/E JCLIN command

You now want to change the target library into which a module distribution library has been totally copied. Assume that the DLIB was MODDLB01, that it was previously copied to LINKLIB, and that it is now copied to LPALIB. You can do this with the following UCL statements:

```

SET    BDY(TGT1)      /* Set to target zone.    */
UCLIN  /*              */
REP    DLIB(MODDLB01) /* Module DLIB.          */
       SYSLIB(LPALIB) /* Now copied to LPALIB. */
       /* No longer in LINKLIB. */
ENDUCL /*              */

```

If the modules should **also** be copied into LPALIB, the following UCL can be used:

```
SET      BDY(TGT1)          /* Set to target zone.      */.  
UCLIN                    /*                               */.  
ADD      DLIB(MODDLB01)     /* Module DLIB.             */.  
          SYSLIB(LPALIB)    /* Now copied to LPALIB.    */.  
          /* SYSLIB added, not replaced.*/.  
ENDUCL                    /*                               */.
```

In both cases, you must take care of copying the modules from the distribution library. SMP/E does not process elements when it records UCL changes.

Now that the UCL changes are made, when SMP/E installs SYSMODs that update MODDLB01, SMP/E checks the DLIB entry for MODDLB01, determines that the modules are copied to LPALIB, and creates or modifies the LMOD entries (with the same name as the MOD entries) to indicate a copy to LPALIB.

**Note:** This technique works before you actually install any service. If you want to change (not add) another library in the DLIB copy list, you must find all the MOD, MAC, SRC, and LMOD entries with a SYSLIB value matching the old target library, and then change that value using UCLIN. If you do not do this, SMP/E continues to copy entries with the old SYSLIB value to the old library (and also, in some cases, to the new library).

---

## DLIBZONE Entry (Distribution Zone)

The DLIBZONE entry contains information SMP/E uses to process a specific distribution zone and the associated distribution libraries. It is created by UCLIN and must be defined before you can do any other processing for that distribution zone.

### Subentries

These are the subentries for the DLIBZONE entry as they appear in the LIST output:

#### *name*

is the name of the distribution zone. You assign the name when the zone is created.

The name can contain from 1 to 7 alphanumeric characters (A through Z, 0 through 9) or national characters (\$, #, @).

#### **ACCJCLIN**

indicates that JCLIN is to be saved in the distribution zone whenever a SYSMOD containing inline JCLIN is accepted.

The UCL operand is **ACCJCLIN**.

- **ACCJCLIN** should be specified only for zones in which all the products have been installed with SMP/E Release 3, or later.
- To save inline JCLIN for a product at ACCEPT time, you must first accept that product using SMP/E Release 3, or later, and have ACCJCLIN set in the DLIBZONE entry. From then on, you must keep ACCJCLIN set in the DLIBZONE entry. This ensures that any time you accept service for that product, its JCLIN is updated in the distribution zone. For more information, see "Inline JCLIN" on page 36.

#### **OPTIONS**

is the name of the OPTIONS entry in the global zone that should be used when processing this distribution zone. For more information, see "OPTIONS Entry (Global Zone)" on page 741.

The UCL operand is **OPTIONS**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.
- This name can be overridden by using the OPTIONS parameter on the SET command. For more information, see Chapter 21.
- If no OPTIONS entry name is specified, SMP/E uses a set of default utility values when processing this distribution zone. For more information, see "OPTIONS Entry (Global Zone)" on page 741.

#### **RELATED**

is the name of the target zone to which this distribution zone is related. A distribution zone is related to the target zone that was built from these distribution libraries, such as during system generation.

The UCL operand is **RELATED**(*zone*).

- The zone name can contain from 1 to 7 alphanumeric characters.

- Although the entry can be defined without this subentry, you **must** define the subentry before you can install any SYSMODs into the distribution libraries.

### SREL

lists the system releases to be supported within this distribution zone.

The UCL operand is **SREL**(*srel...*).

- The SREL must contain 4 characters, usually 1 alphabetic character followed by 3 numeric characters. These are the SRELs defined by IBM:

System	SREL
MVS	Z038
CICS	C150
NCP	P004
IMS, DB2	P115

- Although the entry can be defined without this subentry, you **must** define the subentry before you can install any SYSMODs in the distribution libraries.

**Note:** Although you can support multiple products with different SREL values from one distribution zone, those products are still subject to all other restrictions related to combining products in one zone. The most common reason for not being able to combine products is common element names. For example, modules or macros with the same name are found in both products, but reside in different libraries.

### ZDESC

is a user-written description for this zone.

The UCL operand is **ZONEDESCRIPTION**(*text*).

- The zone description can be in single-byte characters (such as English alphanumeric characters) or in double-byte characters (such as Kanji).
- The zone description can contain up to 500 bytes of data, including blanks. (For double-byte data, the 500-byte maximum includes all shift-in and shift-out characters, as well as the double-byte characters.) Extra blanks are deleted. All data beyond column 72 is ignored, including blanks.
- The zone description cannot be only blanks.
- If parentheses are included in the text, they must be in matched pairs.

## LIST Examples

To list the DLIBZONE entry for a particular distribution zone, you can use the following commands:

```
SET      BDY(DLIB1)          /* Set to requested DLIB. */
LIST     DLIBZONE           /* List DLIBZONE entry.   */
```

Figure 104 on page 677 is an example of LIST output for a DLIBZONE entry.



```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SEMPLIST OUTPUT
DLIB1      DZONE ENTRY

NAME

DLIB1      DZONE          = DLIB1
           ZDESC          = ZONE DESCRIPTION FOR DLIB1 ZONE
           RELATED        = TGT2
           SREL            = Z038
           OPTIONS        = OPTDLIB2
           ACCJCLIN

```

Figure 104. DLIBZONE Entry: Sample LIST Output

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the DLIBZONE entry. When you use UCLIN to update a DLIBZONE entry, remember that if a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.

The following examples are provided to help you use the DLIBZONE entry:

- “Example 1: Defining a DLIBZONE Entry”
- “Example 2: Formatting a Zone Description” on page 678

### Example 1: Defining a DLIBZONE Entry

Assume you are about to add a new function to your system and want to define a separate distribution zone for it named DLIB2. The OPTIONS entry you plan to use is OPTDLIB2, and the SREL for the SYSMOD is Z038. The distribution zone is in data set SMPE.SMPCSI.CSI. After installing the function in the distribution libraries, you plan to do a system generation to build a set of target libraries. The target zone for those libraries is TGT2. The following UCLIN can be used to define the new zone.

```

SET      BDY(GLOBAL)          /* Set to global zone.      */
UCLIN    /* UCLIN for GZONE entry */
ADD      GZONE                /* to set up                */
          ZONEINDEX(          /* index for new zone.      */
                (DLIB2,SMPE.SMPCSI.CSI,DLIB) /*
                )          /*
          /*
          /*
ENDUCL   /* End global zone update. */

SET      BDY(DLIB2)           /* Now define new zone.     */
UCLIN    /* UCLIN to define it.    */
ADD      DLIBZONE(DLIB2)      /* Identify name.          */
          OPTIONS(OPTDLIB2)   /* OPTIONS entry to use.   */
          SREL(Z038)          /* SREL for MVS.           */
          RELATED(TGT2)       /* For this tgt library.   */
          ZDESC(
                THIS DISTRIBUTION ZONE IS FOR
                SREL Z038
          )          /* Zone description.      */
          /*
          /*
ENDUCL   /*

```

**Note:** Even if the OPTIONS entry or TARGETZONE entry has not yet been defined, you can still refer to it in the DLIBZONE entry. However, you must create the entry before you process this distribution zone. For examples of defining OPTIONS entries, see "OPTIONS Entry (Global Zone)" on page 741. For examples of defining TARGETZONE entries, see "TARGETZONE Entry (Target Zone)" on page 782.

## Example 2: Formatting a Zone Description

Assume you enter the following zone description with the first line ending in column 72 and the second line starting in column 1:

```

-----+-----1-----+-----2----- ... -----5-----+-----6-----+-----7--
SET      BDY(DLIB1)          /* Set to DLIB zone.      */.
UCLIN    /* UCLIN for DZONE entry */.
ADD      DZONE(DLIB1)       /* to set up.             */.
          ZDESC(             THIS IS THE DESCRIPTION FOR
THE DLIB1 ZONE)
                                          /* End of zone description. */.
ENDUCL   /* End DLIB zone update.  */.
    
```

Because there is no blank between the word ending in column 72 and the next word starting in column 1, SMP/E runs the two together.

The words in a zone description, even words that end in column 72, must be separated by blanks. To format the zone description in this example correctly, you can put a blank at the beginning of the second line:

```

-----+-----1-----+-----2----- ... -----5-----+-----6-----+-----7--
SET      BDY(DLIB1)          /* Set to DLIB zone.      */.
UCLIN    /* UCLIN for DZONE entry */.
ADD      DZONE(DLIB1)       /* to set up.             */.
          ZDESC(             THIS IS THE DESCRIPTION FOR
THE  DLIB1 ZONE)
                                          /* End of zone description. */.
ENDUCL   /* End DLIB zone update.  */.
    
```

Because there is a blank explicitly coded between the word ending in column 72 and the word starting in column 1, SMP/E does not run the words together.

## FMIDSET Entry (Global Zone)

The FMIDSET entry defines a group of FMIDs for use in limiting the SYSMODs processed by an SMP/E command. For example, you can specify an FMIDSET on the FORFMID operand of the APPLY command to process only SYSMODs applicable to one of the FMIDs in the FMIDSET.

### Subentries

These are the subentries for the FMIDSET entry as they appear in the LIST output:

*name*

is the name of the FMIDSET.

The name can contain from 1 to 8 alphanumeric characters.

**FMID**

lists the function SYSMODs (that is, FMIDs) that are to be part of this FMIDSET.

The UCL operand is **FMID(sysmod\_id...)**.

The SYSMOD ID must contain 7 alphanumeric characters.

### LIST Examples

To list all the FMIDSET entries in a global zone, you can use the following commands:

```
SET      BDY(GLOBAL)      /* FMIDSET in global only. */
LIST     FMIDSET          /* list all FMIDSET entries. */
```

To list specific FMIDSET entries in a global zone, you can use these commands:

```
SET      BDY(GLOBAL)      /* FMIDSET in global only. */
LIST     FMIDSET(ALLSET  /* List only these two */
           XXSET)        /* entries. */
```

The format of the LIST output for each FMIDSET entry is the same for both of these commands. The only difference is the number of FMIDSET entries listed.

Figure 105 is an example of LIST output for FMIDSET entries.

PAGE <i>nnnn</i> - NOW SET TO <i>zzzzzz</i> ZONE <i>nnnnnnn</i> DATE <i>mm/dd/yy</i> TIME <i>hh:mm:ss</i> GIMSMP LVL 18.1. <i>nn</i> SMPLIST OUTPUT						
GLOBAL	FMIDSET ENTRIES					
	NAME					
ALLSET	FMID	=	FXX1102 JXX1121 JXX1122 JXX1123			
			FYY1102 JYY1121 JYY1122 JYY1123			
			FZZ1102 JZZ1121 JZZ1122 JZZ1123			
XXSET	FMID	=	FXX1102 JXX1121 JXX1122 JXX1123			
YYSET	FMID	=	FYY1102 JYY1121 JYY1122 JYY1123			
ZZSET	FMID	=	FZZ1102 JZZ1121 JZZ1122 JZZ1123			

Figure 105. FMIDSET Entry: Sample LIST Output

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in an FMIDSET entry. When you use UCLIN to update an FMIDSET entry, keep these points in mind:

- After the UCLIN changes are made, the FMIDSET entry must contain at least an FMID subentry. Otherwise there is not enough information in the entry for SMP/E to use the entry.
- If a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.

The following examples are provided to help you use the FMIDSET entry:

- "Example 1: Defining an FMIDSET Entry"
- "Example 2: Modifying an FMIDSET Entry"

### Example 1: Defining an FMIDSET Entry

Assume you have four function SYSMODs installed, JXX1102, JXX1121, JXX1122, and JXX1123, and that when installing service you would like to be able to install just the PTFs for these products. You could set up an FMIDSET entry as follows:

```
SET      BDY(GLOBAL)          /* FMIDSETs only in global. */.  
UCLIN                                /*                               */.  
ADD      FMIDSET(XXSET)       /* Define FMIDSET name.      */  
          FMID(FXX1102        /* Define function SYSMODs  */  
            JXX1121          /* to be part of set.       */  
            JXX1122          /*                               */  
            JXX1123)         /*                               */  
          /*                               */.  
ENDUCL                                /*                               */.
```

You can use the FORFMID operand on the APPLY as follows:

```
SET      BDY(TGT1)           /* Set for APPLY to TGT1.    */.  
APPLY    FORFMID(XXSET)     /* Apply XX PTFs.           */.
```

### Example 2: Modifying an FMIDSET Entry

Once an FMIDSET has been defined, it can be modified as new function SYSMODs are installed or as existing function SYSMODs are replaced. For example, starting with the FMIDSET from the previous example, assume you install a new feature, JXX1124, that deletes JXX1121, and that you want to modify the FMIDSET definition. You can do this as follows:

```
SET      BDY(GLOBAL)          /* FMIDSETs only in global. */.  
UCLIN                                /*                               */.  
REP      FMIDSET(XXSET)       /* Define FMIDSET name.      */  
          FMID(FXX1102        /* Define function SYSMODs.  */  
            /* JXX1121 left out.   */  
            JXX1122          /*                               */  
            JXX1123          /*                               */  
            JXX1124)         /* JXX1124 added.           */  
          /*                               */.  
ENDUCL                                /*                               */.
```

You can also do the following:

```
SET      BDY(GLOBAL)      /* FMIDSETs only in global. */.  
UCLIN                               /*                               */.  
DEL      FMIDSET(XXSET)   /* Define FMIDSET name.     */  
          FMID(JXX1121)   /* Delete JXX1121 from set. */  
                               /*                               */.  
ADD      FMIDSET(XXSET)   /* Define FMIDSET name.     */  
          FMID(JXX1124)   /* Add new FMID.            */  
                               /*                               */.  
ENDUCL                               /*                               */.
```

The result in both cases is the same. The choice of which method to use most likely depends on the number of FMIDs defined in the FMIDSET.

### GLOBALZONE Entry (Global Zone)

The GLOBALZONE entry contains processing-related information for SMP/E. It is also used by SMP/E as an index to target and distribution zones, either in the same CSI or a different CSI data set. The GLOBALZONE entry is created by UCLIN and must be defined before you can do any other processing for that global zone.

### Subentries

These are the subentries for the GLOBALZONE entry as they appear in the LIST output:

#### FMID

lists the function SYSMODs for which SMP/E is to receive service.

The UCL operand is **FMID**(*sysmod\_id...*).

The SYSMOD ID must contain 7 alphanumeric characters.

#### OPTIONS

is the name of the OPTIONS entry in the global zone that should be used in processing this global zone. For more information, see "OPTIONS Entry (Global Zone)" on page 741.

The UCL operand is **OPTIONS**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.
- This name can be overridden by using the OPTIONS parameter on the SET command. For more information, see Chapter 21.
- If no OPTIONS entry name is specified, SMP/E uses a set of default utility values when processing the global zone. For more information, see "OPTIONS Entry (Global Zone)" on page 741.

#### SREL

lists the system releases to be supported in this global zone.

The UCL operand is **SREL**(*srel...*).

The SREL must contain 4 characters, usually 1 alphabetic character followed by three numerics. These are the SRELs defined by IBM:

System	SREL
MVS	Z038
CICS	C150
NCP	P004
IMS, DB2	P115

**Note:** Although you can support multiple products with different SREL values from one global zone, those products are still subject to all other restrictions related to combining products in one zone. The most common reason for not being able to combine products is common element names. For example, modules or macros with the same name are found in both products, but reside in different libraries.

**ZDESC**

is a user-written description for this zone.

The UCL operand is **ZONEDESCRIPTION**(*text*).

- The zone description can be in single-byte characters (such as English alphanumeric characters) or in double-byte characters (such as Kanji).
- The zone description can contain up to 500 bytes of data, including blanks. (For double-byte data, the 500-byte maximum includes all shift-in and shift-out characters, as well as the double-byte characters.) Extra blanks are deleted. All data beyond column 72 is ignored, including blanks.
- The zone description cannot be only blanks.
- If parentheses are included in the text, they must be in matched pairs.

**ZONEINDEX**

identifies all the target zones and distribution zones associated with this global zone. This list is used by SMP/E to determine the data set in which the zone resides and the type of zone.

For ADD and REP statements, the UCL operand is **ZONEINDEX**((*name,dsn,type*)...).

For DEL statements, the UCL operand is **ZONEINDEX**((*name*)...).

- **ZONEINDEX** can also be specified as **ZINDEX**.
- *name* is the name of the zone. It can contain from 1 to 7 alphanumeric characters and should begin with an alphabetic character.
- *dsn* is the fully qualified name of the data set in which the zone resides.
- *type* is the zone type, either TARGET or DLIB.

The ZONEINDEX is only a pointer to a zone. Changes you make to a ZONEINDEX do not affect the associated zone. Therefore, you cannot add, rename, or delete a zone by simply adding, replacing, or deleting its ZONEINDEX. However, these are some other commands you can use:

- **Adding a zone:** To add a zone, you can use the following commands, depending on the particular situation:
  - UCLIN to add a ZONEINDEX, then UCLIN to define the DZONE or TZONE entry
  - ZONECOPY
  - ZONERENAME
  - ZONEEXPORT, UCLIN for the ZONEINDEX, and ZONEIMPORT
- **Renaming a zone:** To rename a zone, you must use the following command:
  - ZONERENAME
- **Deleting a zone:** To delete a zone, you can use the following commands, depending on the particular situation:
  - ZONEDELETE
  - ZONEEXPORT

For more information about any of these commands, see the chapter on that command.

### LIST Examples

To list all the GLOBALZONE entry, you can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to global zone.  */.  
LIST     GLOBALZONE       /* List GLOBALZONE entry. */.
```

Figure 106 is an example of LIST output for a GLOBALZONE entry.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT  
GLOBAL      GZONE ENTRY  
  
NAME  
GLOBAL      ZONEINDEX      = TGT1      TARGET  SYS1.TGT.SMPCSI.CSI  
              TGT2      TARGET  SYS1.TGT.SMPCSI.CSI  
              DLIB1     DLIB    SYS1.DLIB.SMPCSI.CSI  
              DLIB2     DLIB    SYS1.DLIB.SMPCSI.CSI  
ZDESC       = ZONE DESCRIPTION FOR GLOBAL ZONE  
SREL        = Z038      I115      P115      R020  
FMID        = JXY1102   FXY1121   FXY1122   FXY1123  
OPTIONS     = OPTGBL
```

Figure 106. GLOBALZONE Entry: Sample LIST Output

### UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the GLOBALZONE entry. When you use UCLIN to update a GLOBALZONE entry, keep these points in mind:

- After the UCLIN changes are made, the GLOBALZONE entry must contain at least one of these subentries: FMID, OPTIONS, SREL, or ZONEINDEX. Otherwise, there is not enough information in the entry for SMP/E to use the entry.
- If a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.

The following examples are provided to help you use the GLOBALZONE entry:

- “Example 1: Defining the GLOBALZONE Entry”
- “Example 2: Deleting Two ZONEINDEX Entries” on page 685
- “Example 3: Formatting a Zone Description” on page 686

#### Example 1: Defining the GLOBALZONE Entry

To perform any SMP/E processing, you must first define the GLOBALZONE entry. Given the following assumptions:

- You are supporting functions JXY1102, JXY1121, JXY1122, and JXY1123 from this global zone.
- You are supporting system releases Z038, P115, and R020, and I121.



- You are eventually going to define two target zones, TGT1 and TGT2, to exist on SYS1.TGT.SMPCSI.CSI, and two distribution zones, DLIB1 and DLIB2, to exist on SYS1.DLIB.SMPCSI.CSI.
- The OPTIONS entry used is OPTGBL.

The following UCLIN can be used to define the GLOBALZONE entry:

```

SET     BDY(GLOBAL)      /* Set to global zone.     */.
UCLIN                               /*                          */.
ADD     GLOBALZONE      /* Define global now.      */.
        SREL(Z038,P115, /* Identify SRELS.       */.
           R020,I115)    /*                          */.
        FMID(JXY1102,   /* Identify functions.    */.
           JXY1121,     /*                          */.
           JXY1122,     /*                          */.
           JXY1123)     /*                          */.
        OPTIONS(OPTGBL) /* OPTIONS entry to be used.*/.
        ZONEINDEX(     /*                          */.
           (TGT1,SYS1.TGT.SMPCSI.CSI,TARGET) /* */.
           (TGT2,SYS1.TGT.SMPCSI.CSI,TARGET) /* */.
           (DLIB1,SYS1.DLIB.SMPCSI.CSI,DLIB) /* */.
           (DLIB2,SYS1.DLIB.SMPCSI.CSI,DLIB) /* */.
        )               /*                          */.
ENDUCL                               /*                          */.

```

**Note:** Even though the OPTIONS entry has not been set up yet, you can still refer to it in the GLOBALZONE entry. The same is true for the target zone name. However, the OPTIONS entry must be created before the global zone is processed. For examples of setting up the OPTIONS entries, see "OPTIONS Entry (Global Zone)" on page 741.

### Example 2: Deleting Two ZONEINDEX Entries

Assume you decide not to create the TGT1 target zone and the DLIB1 distribution zone defined in the ZONEINDEX of example 1. You should delete the entries for those two zones by using UCL as follows:

```

SET     BDY(GLOBAL)      /* Set to global zone.     */.
UCLIN                               /*                          */.
DEL     GLOBALZONE      /* Define global now.      */.
        ZONEINDEX(     /*                          */.
           (TGT1)       /* Delete TGT1 entry.     */.
           (DLIB1)      /* Delete DLIB1 entry.    */.
        )               /*                          */.
ENDUCL                               /*                          */.

```

**Note:** The CSI data set name and the zone type operands are not required for a delete, although the open and close parentheses around the zones are still required.

**Example 3: Formatting a Zone Description**

Assume you enter the following zone description with the first line ending in column 72 and the second line starting in column 1:

```

-----1-----2----- ... -----5-----6-----7--
SET      BDY(GLOBAL)      /* Set to global zone.  */.
UCLIN    /* UCLIN for GZONE entry  */.
ADD      GZONE            /* to set up.          */
          ZDESC(          THIS IS THE DESCRIPTION FOR
THE GLOBAL ZONE)
                                     /* End of zone description. */.
ENDUCL   /* End global zone update. */.
    
```

Because there is no blank between the word ending in column 72 and the next word starting in column 1, SMP/E runs the two together.

The words in a zone description, even words that end in column 72, must be separated by blanks. To format the zone description in this example correctly, you can put a blank at the beginning of the second line:

```

-----1-----2----- ... -----5-----6-----7--
SET      BDY(GLOBAL)      /* Set to global zone.  */.
UCLIN    /* UCLIN for GZONE entry  */.
ADD      GZONE            /* to set up.          */
          ZDESC(          THIS IS THE DESCRIPTION FOR
    THE GLOBAL ZONE)
                                     /* End of zone description. */.
ENDUCL   /* End global zone update. */.
    
```

Because there is a blank explicitly coded between the word ending in column 72 and the word starting in column 1, SMP/E does not run the words together.

---

## HFS Entry (Distribution and Target Zone)

The HFS entry describes an HFS element that exists in a distribution library or a hierarchical file system. An HFS entry is created the first time you install a SYSMOD containing an MCS for an HFS element that does not yet have an entry in the CSI data set.

SMP/E records the function and service level of the HFS element in the entry. Once an HFS entry exists, it is updated as subsequent SYSMODs affecting the HFS element are installed.

### Subentries

These are the subentries for the HFS entry as they appear in the LIST output:

#### *name*

is the name of the HFS element represented by the entry.

The name can contain 1–8 uppercase alphabetic, numeric, or national (\$, #, @) characters.

#### **BINARY or TEXT**

indicates the installation mode to be used when the HFS copy utility is invoked to install the element into the hierarchical file system.

The UCL operand is **BINARY** or **TEXT**.

- *Binary* mode means that the element is installed in its entirety as a data stream, with no breaks for logical records.
- *Text* mode means that the element is installed with breakpoints for logical records.
- **BINARY** and **TEXT** are mutually exclusive.
- If there is no mode indicator in the HFS entry, the HFS copy utility determines how to install the element.

#### **DISTLIB**

specifies the ddname of the distribution library for the HFS element.

The UCL operand is **DISTLIB**(*ddname*).

- The ddname can contain any uppercase alphabetic, numeric, or national (\$, #, @) character, and can be 1–8 characters long.
- The **DISTLIB** subentry is required. Without it, SMP/E cannot process any changes for the HFS element.

#### **FMID**

identifies the functional owner of this HFS element. The functional owner is the last function SYSMOD that replaced this element.

The UCL operand is **FMID**(*sysmod\_id*).

The SYSMOD ID must contain 7 uppercase alphabetic, numeric, or national (\$, #, @) characters.

### LASTUPD

identifies the cause of the last change to this HFS entry.

The UCL operand is **LASTUPD**(*value*). This subentry can contain one of the following values:

#### UCLIN

indicates that the change was made as a result of UCLIN processing.

#### sysmod-id

indicates that the change was made during the installation of the indicated SYSMOD.

The SYSMOD ID must contain 7 uppercase alphabetic, numeric, or national (\$, #, @) characters.

### LASTUPD TYPE

indicates how the entry was last changed.

The UCL operand is **LASTUPDTYPE**(*value*). This subentry can contain one of the following values:

**ADD**      The entry was added.  
**UPD**      The entry was updated.

### LINK

specifies a list of alternative names by which this HFS element can be known in the hierarchical file system.

In LIST output, linknames are always enclosed in single apostrophes. If an apostrophe is part of a linkname, it is always shown as two consecutive apostrophes in LIST output.

The UCL operand is **LINK**(*linkname...*).

- The linkname can be from 1 to 64 characters.
- A linkname can be enclosed in single apostrophes ('). A linkname **must** be enclosed in single apostrophes if any of the following is true:
  - The linkname contains lowercase alphabetic characters.
  - The linkname contains a character that is not uppercase alphabetic, numeric, national (\$, #, or @), slash (/), plus (+), hyphen, period, or ampersand (&).
  - The linkname spans more than one line in the link-edit control statement.

The single apostrophes used to enclose a linkname (the delimiters) do not count as part of the 64-character limit.

- Any apostrophes specified as part of a linkname (not the delimiters) must be doubled.

Double apostrophes count as two characters in the 64-character limit.

- The linkname can include characters X'40' through X'FE'.

### PARM

specifies a character string that is to be passed to the hierarchical file system copy utility as an execution-time parameter. The maximum length of this character string is 300 bytes of nonblank data. If any blanks are specified in the

PARM value, they are deleted by SMP/E during processing and do not count toward the 300-byte maximum.

The UCL operand is **PARM**(*character\_string*).

- PARM is an optional operand.
- The character string can be entered free-form, without regard to blanks (which are compressed out of the string), and can span multiple 80-byte records.
- If parentheses are specified in the PARM value, there must always be a pair (left and right); otherwise, the results are unpredictable.

**RMID**

identifies the last SYSMOD that **replaced** this HFS element. Any subsequent SYSMOD that modifies this HFS element must have defined a relationship (such as PRE or SUP) with this SYSMOD.

The UCL operand is **RMID**(*sysmod\_id*).

- The SYSMOD ID must contain 7 uppercase alphabetic, numeric, or national (\$, #, @) characters.
- If RMID is not specified but **FMID** is, SMP/E sets the RMID value to the specified FMID.

**SYSLIB**

specifies the ddname of the “target library” within the hierarchical file system for the HFS element.

The UCL operand is **SYSLIB**(*ddname*).

- Only one SYSLIB value can be specified.
- The ddname can contain any uppercase alphabetic, numeric, or national (\$, #, @) character, and can be 1 to 8 characters long.
- The SYSLIB subentry is required. Without it, SMP/E cannot process any changes for the HFS element.

**LIST Examples**

To list all the HFS entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
LIST     HFS                /* List all HFS entries.   */.
```

To list specific HFS entries, you can use these commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
LIST     HFS(HFSEL1        /* List only these        */.
         HFSEL2           /* three                  */.
         HFSEL3)          /* entries.               */.
```

The format of the LIST output for each HFS entry is the same for both of these commands. The only difference is the number of HFS entries listed. Figure 107 shows an example of LIST output for HFS entries.

## HFS Entry (Distribution and Target Zone)

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT

TGT1  HFS      ENTRIES

NAME

HFSEL1  LASTUPD      = HFSFUNC TYPE=ADD
        LIBRARIES  = DISTLIB=APOSIXL1  SYSLIB=HFSTGTL1
        BINARY
        FMID       = HFSFUNC
        RMID       = HFSPTF1
        LINK       = 'linkname_1'
                 = 'linkname_2'

HFSEL2  LASTUPD      = HFSFUNC TYPE=ADD
        LIBRARIES  = DISTLIB=APOSIXL2  SYSLIB=HFSTGTL2
        TEXT
        FMID       = HFSFUNC
        RMID       = HFSFUNC
        LINK       = 'linkname_3'
                 = 'linkname_4'

HFSEL3  LASTUPD      = HFSFUNC TYPE=ADD
        LIBRARIES  = DISTLIB=APOSIXL2  SYSLIB=HFSTGTL2
        TEXT
        FMID       = HFSFUNC
        RMID       = HFSPTF2
        LINK       = 'linkname_5'
                 = 'linkname_6'
        PARM       = This is another sample character string specified as the value of PARM. It is a maximum
                    length character string for this subentry. I.e., it is 300 characters long. The string
                    has no blanks in it. If you see something that looks like a blank, it's not. The previous
                    2 characters are 'X'41'..

```

Figure 107. HFS Entry: Sample LIST Output

By specifying the FORFMID operand, you can reduce the number of HFS entries listed. When FORFMID is specified, SMP/E lists an HFS entry only if its FMID matches one of the FMIDs specified on the FORFMID operand. For example, to list HFS entries whose FMIDs either are defined in FMIDSET HFS or are HFSFUNC, you can use these commands:

```

SET      BDY(TGT1)          /* Set to target zone.    */.
LIST     HFS                /* List all HFS entries   */.
        FORFMID(HFS        /* for the HFS FMIDSET   */.
              HFSFUNC)     /* and FMID HFSFUNC.     */.

```

You can use the LIST command to find out the names of all SYSMODs that have modified an HFS element. To include the names of these SYSMODs in the LIST output you can use the XREF operand, as shown in these commands:

```

SET      BDY(TGT1)          /* Set to requested zone. */.
LIST     HFS                /* List all HFS entries   */.
        XREF                /* and related SYSMODs.  */.

```

### Notes:

1. XREF can be used either in mass mode or in select mode.
2. SMP/E obtains the data included for the XREF operand by checking all the SYSMOD entries for subentries for this HFS element. Because this data is not contained in the HFS entry itself, you cannot use UCLIN to change it in the HFS entry.

Figure 108 on page 691 is an example of the LIST output produced when the XREF operand is used.

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
TGT1  HFS      ENTRIES

NAME
HFSEL1  LASTUPD      = HFSFUNC TYPE=ADD
        LIBRARIES   = DISTLIB=APOSIXL1 SYSLIB=HFSTGTL1
        BINARY
        FMID        = HFSFUNC
        RMID        = HFSPTF1
        LINK        = 'linkname_1'
                = 'linkname_2'
        SYSMOD HISTORY = SYSMOD  TYPE      DATE    MCS    --STATUS--
                HFSFUNC  FUNCTION  94.100  HFS    APP    ACC
                HFSPTF1  PTF      94.120  HFS    APP    ACC

HFSEL2  LASTUPD      = HFSFUNC TYPE=ADD
        LIBRARIES   = DISTLIB=APOSIXL2 SYSLIB=HFSTGTL2
        TEXT
        FMID        = HFSFUNC
        RMID        = HFSFUNC
        LINK        = 'linkname_3'
                = 'linkname_4'
        SYSMOD HISTORY = SYSMOD  TYPE      DATE    MCS    --STATUS--
                HFSFUNC  FUNCTION  94.100  HFS    APP    ACC
    
```

Figure 108. HFS Entry: Sample LIST Output When XREF Is Specified

## UNLOAD Examples

To dump the HFS entries in UCL format, you can use the UNLOAD command. To unload all the HFS entries in a particular zone, you can use the following commands:

```

SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD  HFS                 /* Unload all HFS entries. */.
    
```

To unload specific HFS entries, you can use these commands:

```

SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD  HFS(HFSEL1         /* Unload only these      */.
           HFSEL2         /* three                  */.
           HFSEL3)        /* entries.               */.
    
```

The format of the UNLOAD output for each HFS entry is the same for both of these commands. The only difference is the number of HFS entries listed. Figure 109 on page 692 is an example of UNLOAD output for HFS entries.

```

UCLIN .
REP    HFS          ( HFSEL1  )
      LASTUPD      ( HFSFUNC  )
      LASTUPDTYPE  ( ADD )
      DISTLIB      ( APOSIXL1 )
      SYSLIB       ( HFSTGTL1 )
      FMID         ( HFSFUNC  )
      RMID         ( HFSPTF1  )
      LINK         ( 'linkname_1'
                    'linkname_2' )

      .
REP    HFS          ( HFSEL2  )
      LASTUPD      ( HFSFUNC  )
      LASTUPDTYPE  ( ADD )
      DISTLIB      ( APOSIXL2 )
      SYSLIB       ( HFSTGTL2 )
      FMID         ( HFSFUNC  )
      RMID         ( HFSFUNC  )
      LINK         ( 'linkname_3'
                    'linkname_4' )

      .
REP    HFS          ( HFSEL3  )
      LASTUPD      ( HFSFUNC  )
      LASTUPDTYPE  ( ADD )
      DISTLIB      ( APOSIXL2 )
      SYSLIB       ( HFSTGTL2 )
      TEXT
      FMID         ( HFSFUNC  )
      RMID         ( HFSPTF2  )
      LINK         ( 'linkname_5'
                    'linkname_6' )
      PARM         ( This_is_another_sample_character_string_spec
                    ified_as_the_value_of_PARM._It_is_a_maximum
                    ___length_character_string_for_this_subentr
                    y._I.e.,_it_is_300_characters_long._The_st
                    ring___has_no_blanks_in_it._If_you_see_som
                    ething_that_looks_like_a_blank,_it's_not. T
                    he_previous_2_characters_are_X'41'..
                    )

      .
ENDUCL.

```

Figure 109. HFS Entry: Sample UNLOAD Output

By specifying the FORFMID operand, you can reduce the number of HFS entries unloaded. When FORFMID is specified, SMP/E unloads an HFS entry only if its FMID matches one of the FMIDs specified on the FORFMID operand. For example, to unload HFS entries whose FMIDs either are defined in FMIDSET HFS or are HFSFUNC, you can use these commands:

```

SET      BDY(TGT1)          /* Set to target zone.      */.
UNLOAD  HFS                 /* Unload all HFS entries  */.
        FORFMID(HFS        /* for the HFS FMIDSET    */.
              HFSFUNC)     /* and FMID HFSFUNC.     */.

```



## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the HFS entry. After the UCLIN changes are made, the HFS entry must contain at least the following subentries:

- DISTLIB
- FMID
- RMID
- SYSLIB

Otherwise, there is not enough information in the entry to process the HFS element. If any of these subentries are missing, SMP/E does not make the requested UCL updates to the entry, and the entry remains as it was before the UCL command.

The following examples are provided to help you use the HFS entry:

- “Example 1: Adding a New HFS Entry”
- “Example 2: Defining a Linkname for an Existing HFS Element” on page 694

### Example 1: Adding a New HFS Entry

Defining a new HFS entry with UCL is very seldom required; generally, HFS entries are created from the information specified on the ++HFS statements contained in the SYSMODs when SYSMODs are installed. If, however, you want to use UCL in defining a new HFS entry, the following is an example of the minimum information you should provide:

```

SET      BDY(TGT1)          /* Set to target zone.      */
UCLIN                                         /*                          */
ADD      HFS(HFS01)        /* Define new HFS entry.    */
          DISTLIB(AHFSLIB) /* Define DLIB,             */
          SYSLIB(HFSLIB)   /* target library.         */
          FMID(ZUSR001)    /* Functional owner (in this
                          example a user function). */
          RMID(ZUSR001)    /* Same value as FMID.     */
                          /*                          */
ENDUCL                                         /*                          */
SET      BDY(DLB1)        /* Now do same to DLIB.    */
UCLIN                                         /*                          */
ADD      HFS(HFS01)        /* Define new macro entry.  */
          DISTLIB(AHFSLIB) /* Define DLIB,             */
          SYSLIB(HFSLIB)   /* target library.         */
          FMID(ZUSR001)    /* Functional owner (in this
                          example a user function). */
          RMID(ZUSR001)    /* Same value as FMID.     */
                          /*                          */
ENDUCL                                         /*                          */

```

**Note:** If the RMID value had not been specified, it would have defaulted to the FMID value.

### Example 2: Defining a Linkname for an Existing HFS Element

The following shows how to inform SMP/E of new linkname for an existing HFS element:

```

SET      BDY(TGT1)          /* Set to target zone.      */
UCLIN   .                  /*                          */
ADD      HFS(HFS01)        /* Existing HFS entry.     */
        LINK('../user1k') /* New linkname.          */
        .                  /*                          */
ADD      HFS(HFS02)        /* Existing HFS entry.     */
        LINK('myname')   /* New linkname.          */
        .                  /*                          */
ENDUCL  .                  /*                          */
SET      BDY(DLB1)        /* Now do same thing to   */
        .                  /* appropriate DLIB.     */
UCLIN   .                  /*                          */
ADD      HFS(HFS01)        /* Existing HFS entry.     */
        LINK('../user1k') /* New linkname.          */
        .                  /*                          */
ADD      HFS(HFS02)        /* Existing HFS entry.     */
        LINK('myname')   /* New linkname.          */
        .                  /*                          */
ENDUCL  .                  /*                          */

```

**Note:** UCLIN does not create a linkname in the hierarchical file system; that must be done outside of SMP/E using standard utilities. The UCLIN changes ensure that, when the HFS element is subsequently modified, the HFS element is updated under both its primary name and its alternative name in the hierarchical file system.

### Example 3: Adding an HFS Element Entry with a PARM Subentry

If you want to use UCL in adding an HFS element entry with a PARM subentry, the following example shows you how this can be accomplished.

```

SET      BDY(TGT1).        /* Set to target zone.     */
UCLIN   .                  /*                          */
ADD      HFS(HFSEL3) LASTUPD(HFSFUNC) LASTUPDTYPE(ADD)
        DISTLIB(APOSIXL2) SYSLIB(HFSTGTL2) FMID(HFSFUNC) TEXT
        RMID(HFSPTF2) LINK('linkname_5' 'linkname_6' )
        PARM(This_is_another_sample_character_string_specified_as_the_
value_of_PARM._It_is_a_maximum____length_character_string_for_this_
subentry._I.e.,_it_is_300_characters_long._The_string_has_no_
blanks_in_it._If_you_see_something_that_looks_like_a_blank,_it's_not.
The_previous_2_characters_are_X'41'..) .
ENDUCL  .                  /*                          */

```

## HOLDDATA Entry (Global Zone)

The HOLDDATA entry contains ++HOLD statements that either were received from SMPHOLD (external HOLDDATA) or were within a SYSMOD that was received (internal HOLDDATA). It is a record of the ++HOLD statements and is not used by SMP/E to control exception SYSMOD processing. However, when SMP/E saves these ++HOLD statements, it also creates HOLDDATA subentries in the associated global zone SYSMOD entry. SMP/E uses these HOLDDATA subentries to control exception SYSMOD processing.

### LIST Examples

To list all the HOLDDATA entries in a global zone, you can use the following commands:

```
SET    BDY(GLOBAL)      /* Set to requested zone.   */.
LIST   HOLDDATA         /* List all HOLDDATA entries.*/.
```

To list specific HOLDDATA entries in a global zone, you can use these commands:

```
SET    BDY(GLOBAL)      /* Set to requested zone.   */.
LIST   SYSMOD(UZ12345   /* List only these two     */.
        UZ56789)       /* entries                  */.
        HOLDDATA       /* plus HOLDDATA for them. */.
```

The output from both LIST requests differs in the amount of information presented. If you list all the HOLDDATA entries, only the HOLDDATA itself is presented, as in Figure 110 on page 696. If you list specific HOLDDATA entries, the HOLDDATA is included in the SYSMOD entry, as in Figure 111 on page 697.

#### Notes:

1. The ++HOLD statements displayed in the LIST output are exactly as received by SMP/E. They are listed in alphanumeric order by reason ID.
2. If a specified SYSMOD has not yet been received, only the associated HOLDDATA entry is shown. For an example, see SYSMOD UZ56789 in Figure 111 on page 697.
3. HOLDDATA entries for system holds show either *INT* or *EXT* to indicate the source of the ++HOLD statement. *INT* means that the ++HOLD statement was contained in the held SYSMOD and can be resolved only with the BYPASS operand on the APPLY or ACCEPT command. *EXT* means that the ++HOLD statement was obtained from another source, such as SMPHOLD, and can be resolved by either the BYPASS operand or a ++RELEASE statement.
4. You cannot use UCLIN to add, update, or delete HOLDDATA entries. However, you can use the REJECT command to delete HOLDDATA entries. For more information, see Chapter 13.

## HOLDDATA Entry (Global Zone)

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
GLOBAL      HOLDDATA ENTRIES

NAME

UZ12345  HOLDERROR      = AZ00001  ++HOLD(UZ12345) ERROR
          REASON(AZ00001) FMID(HBB2102)
          COMMENT(OC4).
          HOLDERROR      = AZ00002  ++HOLD(UZ12345) ERROR
          REASON(AZ00002) FMID(HBB2102)
          COMMENT(OC4).
          HOLDSYSTEM(INT) = DOC      ++HOLD(UZ12345) SYSTEM
          REASON(DOC) FMID(HBB2102)
          COMMENT(NEW MSG).
          HOLDSYSTEM(EXT) = UCLIN    ++HOLD(UZ12345) SYSTEM
          REASON(UCLIN) FMID(HBB2102)
          COMMENT(UCLIN REQUIRED).
          HOLDUSER       = INUSE     ++HOLD(UZ12345) USER
          REASON(INUSE) FMID(HBB2102)
          COMMENT(IM MODIFYING).

UZ56789  HOLDERROR      = AZ00023  ++HOLD(UZ56789) ERROR
          REASON(AZ00023) FMID(HBB2102)
          COMMENT(OC4).
          HOLDERROR      = AZ00024  ++HOLD(UZ56789) ERROR
          REASON(AZ00024) FMID(HBB2102)
          COMMENT(OC4).
```

Figure 110. HOLDDATA Entry: Sample LIST Output

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
GLOBAL      SYSMOD ENTRIES

NAME

UZ12345  TYPE          = PTF
STATUS    = REC
DATE/TIME REC = 90.100 08:00:00
SREL  VER(001) = Z038
DELETE VER(001) = HBB2102
SUPING VER(001) = AZ11111 AZ11112 AZ11113
MAC      = MAC01  MAC02  MAC03
MOD      = MOD01  MOD02  MOD03  MOD04
SRC      = SRC01  SRC02
HOLD ERROR = AZ00001  ++HOLD(UZ12345) ERROR
                        REASON(AZ00001) FMID(HBB2102)
                        COMMENT(OC4).
HOLDERROR = AZ00002  ++HOLD(UZ12345) ERROR
                        REASON(AZ00002) FMID(HBB2102)
                        COMMENT(OC4).
HOLDSYSTEM(INT) = DOC  ++HOLD(UZ12345) SYSTEM
                        REASON(DOC) FMID(HBB2102)
                        COMMENT(NEW MSG).
HOLDSYSTEM(EXT) = UCLIN ++HOLD(UZ12345) SYSTEM
                        REASON(UCLIN) FMID(HBB2102)
                        COMMENT(UCLIN REQUIRED).
HOLDUSER      = INUSE  ++HOLD(UZ12345) USER
                        REASON(INUSE) FMID(HBB2102)
                        COMMENT(IM MODIFYING).

UZ56789  HOLD ERROR = AZ00023  ++HOLD(UZ56789) ERROR
                        REASON(AZ00023) FMID(HBB2102)
                        COMMENT(OC4).
HOLDERROR = AZ00024  ++HOLD(UZ56789) ERROR
                        REASON(AZ00024) FMID(HBB2102)
                        COMMENT(OC4).

```

Figure 111. HOLDDATA Entry: Sample LIST Output When SYSMOD Is Specified

### LMOD Entry (Distribution and Target Zone)

The LMOD entry contains all the information needed to replace or update a given load module. This includes information such as whether the load module is link-edited or copied during the system generation process, any link-edit statements required to relink the load module, the link-edit attributes of the load module, and the libraries in which it resides. An LMOD entry is generally created by one of the following methods:

- **Installing a SYSMOD that adds the load module.** LMOD entries can be created when a SYSMOD is installed. SMP/E builds an LMOD entry if it encounters a ++MOD statement for a load module that does not yet have an LMOD entry and if the distribution library for the module was totally copied during system generation. For more information about copied load modules, see “DLIB Entry (Distribution and Target Zone)” on page 670 and “Module Replacements” on page 97.
- **Processing JCLIN.** LMOD entries can be created during JCLIN processing when SMP/E scans the copy and link-edit steps. At the same time, SMP/E builds MOD entries for modules that are linked or copied to the load module. For more information, see “Processing” on page 179 and “MOD Entry (Distribution and Target Zone)” on page 723.

### Subentries

These are the subentries for the LMOD entry as they appear in the LIST output:

*name*

is the name of the load module described by the LMOD entry.

The name can contain from 1 to 8 alphanumeric characters.

#### **CALLLIBS**

specifies one or more DDDEF entries, existing in the same zone, that compose the SYSLIB allocation to be used when the load module is link-edited.

The UCL operand is **CALLLIBS**(*name...*).

- The DDDEF names can be from 1 to 8 alphanumeric characters.
- SMP/E does not enforce any limit on the number of names that can be specified in a CALLLIBS subentry list. The actual number of partitioned data sets that can be concatenated depends on the operating system you are running under. Therefore, to make this determination, you must refer to the data management manual for your operating system.
- The order in which the libraries are specified is important because it indicates the order in which the SYSLIB concatenation is built.
- Zones containing LMOD entries with CALLLIBS subentries cannot be processed by SMP/E Release 7 or earlier.

#### **COPY**

is a special SMP/E indicator meaning that the load module was copied during system generation, and that there is a one-to-one correspondence between the distribution library module (the MOD entry) and the target system load module (the LMOD entry). This information is used during APPLY processing to determine whether the LEPARM values from a ++MOD statement are applicable to the load module. It is also used during delete and compress processing to

determine whether a load module can be deleted before the new modules are installed.

The UCL operand is **COPY**.

**LASTUPD**

identifies the cause of the last change to this LMOD entry.

**Note:** If a given UCLIN command specifies only cross-zone subentries, this field is not changed.

The UCL operand is **LASTUPD**(*value*). This subentry can contain one of the following values:

**JCLIN**

indicates that the change was made during JCLIN command processing.

**UCLIN**

indicates that the change was made as a result of UCLIN processing.

*sysmod\_id*

indicates that the change was made during the installation of the indicated SYSMOD.

The SYSMOD ID must contain 7 alphanumeric characters.

**LASTUPD TYPE**

indicates how the entry was last changed.

**Note:** If a given UCLIN command specifies only cross-zone subentries, this field is not changed.

The UCL operand is **LASTUPDTYPE**(*value*). This subentry can contain one of the following values:

- ADD**      The entry was added.
- DEL**      A subentry in the entry was deleted.
- MOV**      The load module was moved.
- REN**      The load module was renamed.
- UPD**      The entry was updated.

**LKED ATTRIBUTES**

identifies the link-edit attributes that must be used when this load module is link-edited. SMP/E supports the following link-edit attributes. For more information, see the reference manual for your link-edit utility.

**AC=1**

specifies that the AC=1 parameter (which is the authorization code) is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **AC=1**.

**ALIGN2**

specifies that the ALIGN2 parameter (alignment on a 2KB boundary) is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **ALIGN2** or **ALN2**.

**AMODE=24**

specifies that the AMODE=24 parameter is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **AMODE=24** or **AMOD=24**.

### **AMODE=31**

specifies that the **AMODE=31** parameter is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **AMODE=31** or **AMOD=31**.

### **AMODE=ANY**

specifies that the **AMODE=ANY** parameter is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **AMODE=ANY** or **AMOD=ANY**.

### **AMODE=MIN**

specifies that the **AMODE=MIN** parameter is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **AMODE=MIN** or **AMOD=MIN**.

### **CALL**

indicates that the **CALL** parameter has been specified for a load module in a **JCLIN** link-edit step. **SMP/E** determines whether to pass **CALL** or **NOCAL** to the link-edit utility based on whether the **LMOD** entry contains a **CALLLIBS** subentry. For more information about link-edit parameters, see “Link-Edit Parameters and Load Module Attributes” on page 97.

The UCL operand is **CALL**.

### **CASE**

specifies that the **CASE** parameter, which controls case sensitivity and folding, is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **CASE(UPPER|MIXED)**.

- **CASE(UPPER)** and **CASE(MIXED)** are mutually exclusive.

### **DC**

specifies that the **DC** parameter, which is the downward compatible attribute, is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **DC**.

### **FETCHOPT**

specifies that the **FETCHOPT** parameter is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **FETCHOPT(PACK | NOPACK, PRIME | NOPRIME)**.

### **MAXBLK**

specifies that the **MAXBLK** parameter is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **MAXBLK(nnnnn)**, where *nnnnn* is a number between 256 and 32760.

### **NE**

specifies that the **NE** parameter, which is the noneditable attribute, is to be passed to the link-edit utility when the load module is link-edited.



The UCL operand is **NE**.

### **NOCALL**

specifies that the NOCALL parameter is to be passed to the link-edit utility when the load module is link-edited. SMP/E determines whether to pass CALL or NOCAL to the link-edit utility based on whether the LMOD entry contains a CALLLIBS subentry. For more information about link-edit parameters, see "Link-Edit Parameters and Load Module Attributes" on page 97.

The UCL operand is **NOCALL** or **NCAL**.

### **OL**

specifies that the OL parameter is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **OL**.

### **OVLY**

specifies that the OVLY parameter, which specifies that the load module is in overlay structure, is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **OVLY**.

### **REFR**

specifies that the REFR parameter, which is the refreshable attribute, is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **REFR**.

### **RENT**

specifies that the RENT parameter, which indicates that the load module is re-entrant, is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **RENT**.

### **REUS**

specifies that the REUS parameter, which indicates that the load module is reusable, is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **REUS**.

### **RMODE=24**

specifies that the RMODE=24 parameter is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **RMODE=24** or **RMOD=24**.

### **RMODE=ANY**

specifies that the RMODE=ANY parameter is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **RMODE=ANY** or **RMOD=ANY**.

### **SCTR**

specifies that the SCTR parameter, which indicates that the load module can be scatter-loaded, is to be passed to the link-edit utility when the load module is link-edited.

The UCL operand is **SCTR**.

### **STD**

is a special SMP/E indication that the load module should be link-edited with none of the above attributes.

STD appears in LIST and UNLOAD output in these cases:

- STD was specified on a UCL statement for this entry.
- No link-edit attributes were specified for this entry. SMP/E uses the standard system link-edit parameters.
- The LMOD entry was created by a JCLIN copy step.

If this attribute is specified when a load module is to be link-edited and no UTILITY entry is being used, SMP/E passes the default link-edit parameters to the link-edit utility. For more information about link-edit parameters, see “Link-Edit Parameters and Load Module Attributes” on page 97.

The UCL operand is **STD**.

### **LKED CONTROL**

contains all the link-edit control cards necessary to relink-edit this load module.

**Note:** Relink-editing a load module is triggered by a replacement for one of the modules within that load module. SMP/E performs only CSECT replaces when relink-editing the load module; it does not completely rebuild the load module from the distribution libraries. Therefore, it is not necessary to save all the link-edit control statements in the LMOD entry; for example, the INCLUDE statements are not saved.

The UCL operands are **++LMODIN** and **++ENDLMODIN**.

- The ++LMODIN and ++ENDLMODIN statements must start in column 1.
- No other operands can start on the same line as the ++LMODIN statement.
- The ++ENDLMODIN statement must be specified if the ++LMODIN statement is specified.
- The link-edit control statements must follow SMP/E's coding conventions. For more information, see “General JCLIN Coding Conventions” on page 180.
- Comment statements are ignored and are not saved in the LMOD entry. (Any link-edit control statement with an asterisk in column 1 is considered a comment statement.)

### **MODDEL**

specifies the modules that were once part of this load module but have been deleted. If the modules are reinstalled, they are linked back into this load module.

The UCL operand is **MODDEL**(*module...*).

The module names can contain from 1 to 8 alphanumeric characters.

### **SYSTEM LIBRARIES**

specifies the target system libraries in which this load module resides.

The UCL operand is **SYSLIB**(*ddname...*).

- There can be up to two SYSLIB values in an LMOD entry.
- The ddnames can contain from 1 to 8 alphanumeric characters.

### XZMOD

specifies one or more modules that were added to the load module by the LINK command. The name of the zone supplying each module is also indicated. Any zone updated with the LINK command or cross-zone information **cannot** be processed by SMP/E Release 6, or earlier.

The UCL operand is **XZMOD**(*module,zone*...).

- The zone name specified for an XZMOD subentry cannot match the name of the set-to zone.
- The XZMOD subentry is added to an LMOD entry automatically during LINK command processing. However, it is almost never removed automatically, except during JCLIN processing. For more information, see “Cross-Zone Relationships” on page 172.
- An LMOD entry can contain XZMOD and XZMODP subentries without any other subentries.
- If UCLIN is used to update an existing LMOD entry and **only** cross-zone subentries (XZMOD and XZMODP) are changed, SMP/E does not update the LASTUPD and LASTUPDTYPE subentries.

### XZMODP

indicates that the load module contains one or more modules from another zone and that XZMOD subentries exist in this LMOD entry.

The UCL operand is **XZMODP**.

- You never need to specify the XZMODP subentry on a UCL statement. SMP/E automatically determines the setting of XZMODP, according to whether the LMOD entry contains XZMOD subentries.
- You cannot add the XZMODP subentry to an LMOD entry that does not contain XZMOD subentries.
- You cannot delete the XZMODP subentry from an LMOD entry containing XZMOD subentries.
- An LMOD entry can contain XZMOD and XZMODP subentries without any other subentries.
- If UCLIN is used to update an existing LMOD entry and **only** cross-zone subentries are changed (XZMOD and XZMODP), SMP/E does not update the LASTUPD and LASTUPDTYPE subentries.

## LIST Examples

To list all the LMOD entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
LIST     LMOD              /* List all LMOD entries. */.
```

To list specific LMOD entries, you can use these commands:

## LMOD Entry (Distribution and Target Zone)

```
SET      BDY(TGT1)          /* Set to requested zone. */.
LIST     LMOD(LMOD01       /* List only these two    */.
          LMOD02)         /* entries.                */.
```

The format of the LIST output for each LMOD entry is the same for both of these commands. The only difference is the number of LMOD entries listed. Figure 112 is an example of LIST output for LMOD entries.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
TGT1  LMOD  ENTRIES

NAME

BPXLMOD1 LASTUPD      = HBPX001 TYPE=ADD
          SYSTEM LIBRARY = BPXLIB1
          LKED ATTRIBUTES = RENT      CASE(MIXED)
          LKED CONTROL   = ENTRY BPXMOD01
                          ALIAS ../FRIENDLY/NAME/THAT/DOES/NOT/NEED/QUOTES
                          ALIAS '../friendly/name/with a comma,/which/requires/quotes'
                          ALIAS '../friendly/name/with/a/''/which/requires/quotes'

LMOD01  LASTUPD      = JXY1102 TYPE=ADD
          SYSTEM LIBRARY = LINKLIB
          LKED ATTRIBUTES = STD
          MODDEL         = MOD05      MOD06
          LKED CONTROL   = ENTRY MOD01
                          ALIAS LMOD01A1

LMOD02  LASTUPD      = JXY1102 TYPE=ADD
          SYSTEM LIBRARY = LINKLIB PPLIB01
          LKED ATTRIBUTES = RENT      REUS      AC=1

LMOD03  LASTUPD      = JXY1102 TYPE=ADD
          SYSTEM LIBRARY = LINKLIB
          LKED ATTRIBUTES = RENT      REUS      AC=1
          LKED CONTROL   = ALIAS MOD03
                          *** CHANGE/REPLACE STMTS FOR MOD03 FROM DLIB AOS12
                          CHANGE MOD03C1(NEW03C1)
                          CHANGE MOD03C2(NEW03C2)
```

Figure 112 (Part 1 of 2). LMOD Entry: Sample LIST Output

LMOD04	LASTUPD	= LINK	TYPE=UPD		
	SYSTEM LIBRARY	= LINKLIB			
	LKED ATTRIBUTES	= STD			
	XZMODP				
	XZMOD	= CICSMOD1	FROM ZONE CICS1		
		= CICSMOD2	FROM ZONE CICS1		
		= CICSMOD4	FROM ZONE CICS1		
		= IMSMOD1	FROM ZONE IMS1		
	LKED CONTROL	= ENTRY MOD04			
		ALIAS LMOD04A1			
LMOD05	XZMODP				
	XZMOD	= CICSMOD1	FROM ZONE CICS1		
		= CICSMOD2	FROM ZONE CICS1		
		= IMSMOD1	FROM ZONE IMS1		
LMOD06	LASTUPD	= JCLIN	TYPE=ADD		
	SYSTEM LIBRARY	= LINKLIB			
	LKED ATTRIBUTES	= SCTR	MAXBLK(6160)	AC=1	AMOD=MIN
LMOD07	LASTUPD	= JCLIN	TYPE=ADD		
	SYSTEM LIBRARY	= LINKLIB			
	LKED ATTRIBUTES	= FETCHOPT(PACK,PRIME)	NOCALL		
	MODDEL	= ISPLINK	MOD01		
LMOD08	LASTUPD	= HXY0001	TYPE=ADD		
	SYSTEM LIBRARY	= APPLLIB			
	LKED ATTRIBUTES	= RENT	REUS		
	CALLLIBS	= PLIBASE	CSLIB		
MODULE1	LASTUPD	= UCLIN	TYPE=ADD		
	SYSTEM LIBRARY	= LINKLIB			
	LKED ATTRIBUTES	= STD			
	MODDEL	= MOD1	MOD2		
MODUL25	LASTUPD	= UCLIN	TYPE=UPD		
	SYSTEM LIBRARY	= LINKLIB			
	LKED ATTRIBUTES	= STD			

Figure 112 (Part 2 of 2). LMOD Entry: Sample LIST Output

You can use the LIST command to find out the names of all the distribution library modules that are link-edited into this load module. To include the names of these modules in the LIST output you can use the XREF operand, as shown in these commands:

```

SET      BDY(TGT1)          /* Set to requested zone.  */
LIST     LMOD               /* List all LMOD entries   */
        XREF               /* and MODs in them.     */
    
```

**Notes:**

1. XREF can be used either in mass mode or in select mode.
2. SMP/E obtains the data included for the XREF operand by checking the LMOD subentries in all the MOD entries. Because this data is not contained in the LMOD entry itself, you cannot use UCLIN to change it in the LMOD entry.

## LMOD Entry (Distribution and Target Zone)

Figure 113 is an example of the LIST output produced when the XREF operand is used.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT

TGT1  LMOD  ENTRIES

NAME

BPXLMOD1  LASTUPD      = HBPX001 TYPE=ADD
          SYSTEM LIBRARY = BPXLIB1
          LKED ATTRIBUTES = RENT      CASE(MIXED)
          LKED CONTROL   = ENTRY BPXMOD01
                        ALIAS ../FRIENDLY/NAME/THAT/DOES/NOT/NEED/QUOTES
                        ALIAS '../friendly/name/with a comma,/which/requires/quotes'
                        ALIAS '../friendly/name/with/a/'/which/requires/quotes'
          MODULES       = NAME      FMID
                        BPXMOD01  HBPX001

LMOD01    LASTUPD      = JCLIN    TYPE=ADD
          SYSTEM LIBRARY = SLIB01
          LKED ATTRIBUTES = COPY     RENT      REUS      SCTR      OVLY      EFR      DC      NE      AC=1
                        ALIGN2    AMODE=24  RMODE=24
          MODULES       = NAME      FMID
                        MODULE2    HXZ1234

LMOD02    LASTUPD      = JCLIN    TYPE=ADD
          SYSTEM LIBRARY = SLIB02
          LKED ATTRIBUTES = COPY     RENT      REUS      SCTR      OVLY      EFR      DC      NE      AC=1
                        ALIGN2    AMODE=24  RMODE=24
          MODULES       = NAME      FMID
                        MODULE2    HXZ1234

LMOD03    LASTUPD      = JXY1102 TYPE=ADD
          SYSTEM LIBRARY = LINKLIB
          LKED ATTRIBUTES = RENT     REUS      AC=1
          LKED CONTROL   = ALIAS MOD3
                        *** CHANGE/REPLACE STMTS FOR MOD03 FROM DLIB A0S12
                        CHANGE MOD03C1(NEW03C1)
                        CHANGE MOD03C2(NEW03C2)
          MODULES       = NAME      FMID
                        MOD03     JXY1121
```

Figure 113 (Part 1 of 2). LMOD Entry: Sample LIST Output When XREF Is Specified

LMOD04	LASTUPD	= LINK	TYPE=UPD		
	SYSTEM LIBRARY	= LINKLIB			
	LKED ATTRIBUTES	= STD			
	XZMODP				
	XZMOD	= CICSMOD1 FROM ZONE CICS1			
		= CICSMOD2 FROM ZONE CICS1			
		= CICSMOD4 FROM ZONE CICS1			
		= IMSMOD1 FROM ZONE IMS1			
	LKED CONTROL	= ENTRY MOD04			
		= ALIAS LMOD04A1			
	MODULES	= NAME FMID			
		= MODULE4 HXZ1234			
LMOD05	XZMODP				
	XZMOD	= CICSMOD1 FROM ZONE CICS1			
		= CICSMOD2 FROM ZONE CICS1			
		= IMSMOD1 FROM ZONE IMS1			
LMOD06	LASTUPD	= JCLIN	TYPE=ADD		
	SYSTEM LIBRARY	= LINKLIB			
	LKED ATTRIBUTES	= SCTR	MAXBLK(6160)	AC=1	AMOD=MIN
	MODULES	= NAME FMID			
		= MODULE5 HXZ1234			
LMOD07	LASTUPD	= JCLIN	TYPE=ADD		
	SYSTEM LIBRARY	= LINKLIB			
	LKED ATTRIBUTES	= FETCHOPT(PACK,PRIME)	NOCALL		
	MODEL	= ISPLINK	MOD01		
	MODULES	= NAME FMID			
		= MODULE6 JXY0001			
LMOD08	LASTUPD	= HXY0001	TYPE=ADD		
	SYSTEM LIBRARY	= APPLLIB			
	LKED ATTRIBUTES	= RENT	REUS		
	CALLLIBS	= PLIBASE	CSSLIB		
	MODULES	= NAME FMID			
		= MODULE7 HXY0001			
		= MODULE8 HXY0001			
MODULE1	LASTUPD	= UCLIN	TYPE=UPD		
	SYSTEM LIBRARY	= LINKLIB			
	LKED ATTRIBUTES	= STD			
	MODEL	= MOD1	MOD2		
	MODULES	= NAME FMID			
		= MODULE1 ???????			
MODUL25	LASTUPD	= UCLIN	TYPE=UPD		
	SYSTEM LIBRARY	= LINKLIB			
	LKED ATTRIBUTES	= STD			
	MODULES	= NAME FMID			
		= MODUL25 HXZ1234			

Figure 113 (Part 2 of 2). LMOD Entry: Sample LIST Output When XREF Is Specified

## UNLOAD Examples

To dump the LMOD entries in UCL format, you can use the UNLOAD command. To unload all the LMOD entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD  LMOD               /* Unload all LMOD entries. */.
```

## LMOD Entry (Distribution and Target Zone)

To unload specific LMOD entries, you can use these commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD  LMOD(LMOD01        /* Unload only these two */
        LMOD02)           /* entries.                */.
```

The format of the UNLOAD output for each LMOD entry is the same for both of these commands. The only difference is the number of LMOD entries listed.

Figure 114 is an example of UNLOAD output for LMOD entries.

```
UCLIN .
REP      LMOD              ( LMOD01  )
        LASTUPD           ( JXY1102 )
        LASTUPDTYPE      ( ADD  )
        SYSLIB            ( LINKLIB  )
        /* LEPARM        */ STD
        MODEL             ( MOD05   MOD06  )

++LMODIN
ENTRY MOD01
ALIAS LMOD01A1
++ENDLMODIN

REP      LMOD              ( LMOD02  )
        LASTUPD           ( JXY1102 )
        LASTUPDTYPE      ( ADD  )
        SYSLIB            ( LINKLIB  PPLIB01 )
        /* LEPARM        */ RENT     REUS     AC=1

REP      LMOD              ( LMOD03  )
        LASTUPD           ( JXY1102 )
        LASTUPDTYPE      ( ADD  )
        SYSLIB            ( LINKLIB  )
        /* LEPARM        */ RENT     REUS     AC=1

++LMODIN
ALIAS MOD03
CHANGE MOD03C1(NEW03C1)
CHANGE MOD03C2(NEW03C2)
INCLUDE AOS12 (MOD03  )
++ENDLMODIN
```

Figure 114 (Part 1 of 3). LMOD Entry: Sample UNLOAD Output



```

REP      LMOD          ( LMOD04  )
          LASTUPD      ( LINK    )
          LASTUPDTYPE  ( UPD     )
          SYSLIB       ( LINKLIB  )
          /* LEPARM    */ STD
          /* CROSS-ZONE*/ XZMODP
          XZMOD        (
                        ( CICSMOD1 CICS1 )
                        ( CICSMOD2 CICS1 )
                        ( IMSMOD1  IMS1  )
                        )
++LMODIN
  ENTRY MOD01
  ALIAS LMOD04A1
++ENDLMODIN

REP      LMOD          ( LMOD05  )
          /* CROSS-ZONE*/ XZMODP
          XZMOD        (
                        ( CICSMOD1 CICS1 )
                        ( CICSMOD2 CICS1 )
                        ( IMSMOD1  IMS1  )
                        )

REP      LMOD          ( LMOD06  )
          LASTUPD      ( JCLIN   )
          LASTUPDTYPE  ( ADD     )
          SYSLIB       ( LINKLIB  )
          /* LEPARM    */ /* SCTR   MAXBLK(6160)  AC=1  AMOD=MIN

REP      LMOD          ( LMOD07  )
          LASTUPD      ( JCLIN   )
          LASTUPDTYPE  ( ADD     )
          SYSLIB       ( LINKLIB  )
          MODDEL       ( ISPLINK  MOD01 )
          /* LEPARM    */ /* FETCHOPT(PACK,PRIME)  NOCALL

REP      LMOD          ( LMOD08  )
          LASTUPD      ( HXY0001 )
          LASTUPDTYPE  ( ADD     )
          SYSLIB       ( APPLLIB  )
          /* LEPARM    */ /* RENT   REUS
          CALLLIBS     ( PLIBASE  CSSLIB )

```

Figure 114 (Part 2 of 3). LMOD Entry: Sample UNLOAD Output

```
REP      LMOD          ( MOD04   )
          LASTUPD      ( JXY1102 )
          LASTUPDTYPE  ( ADD     )
          SYSLIB       ( LPALIB  )
          /* LEPARM    */ COPY   STD
          .
ENDUCL.
```

Figure 114 (Part 3 of 3). LMOD Entry: Sample UNLOAD Output

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the LMOD entry. When you use UCLIN to update an LMOD entry, keep these points in mind:

- After the UCLIN changes are made, the LMOD entry must contain at least a SYSLIB subentry. Otherwise, there is not enough information in the entry to indicate where the load module should be installed.
- The input following the ++LMODIN statement replaces all existing link-edit control cards in the LMOD entry. This is different from JCLIN processing, where all control cards are replaced except CHANGE and REPLACE, which are merged with the existing CHANGE and REPLACE control cards.
- If a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.
- When SMP/E processes a DEL statement, it does not compare any control cards after the ++LMODIN statement with the control cards that are currently in the LMOD entry. It deletes all the existing control cards.

The following examples are provided to help you use the LMOD entry:

- “Example 1: Adding a New LMOD Entry” on page 711
- “Example 2: Changing the Link-Edit Attributes of an LMOD” on page 711
- “Example 3: Deleting a MODDEL Subentry” on page 712
- “Example 4: Completing Cross-Zone Updates” on page 712
- “Example 5: Adding a CALLLIBS Subentry List to an LMOD Entry” on page 713
- “Example 6: Deleting Link-Edit Control Statements” on page 713

### Example 1: Adding a New LMOD Entry

The correct method of adding a new LMOD entry is through JCLIN, so that in addition to the LMOD entry being created, SMP/E also ensures that all related MOD entries are updated. If, however, you want to define a new load module entry using UCL, the following is an example of the minimum information you should provide.

```

SET      BDY(TGT1)          /* Set to target zone.    */
UCLIN                                /*                          */
ADD      LMOD(LMOD01)       /* Define new LMOD entry. */
          SYSLIB(LPALIB)    /* System library 1.      */
          PPLIB01)         /* System library 2.      */
          RENT REUS AC=1    /* All link attributes.   */
++LMODIN                          /* All link-edit stmts.  */
  INCLUDE AOS12(MOD01)
  INCLUDE AOS12(MOD02)
  ENTRY  MOD01
  ALIAS  LMOD01A1
  NAME   LMOD01(R)
++ENDLMODIN

REP      MOD(MOD01)         /* Now fix MOD entries.   */
          DISTLIB(AOS12)   /* MOD DLIB.              */
          FMID(FXY1102)    /* Functional owner.      */
          LMOD(LMOD01)     /* Connect to LMOD.       */
          /*                          */
REP      MOD(MOD02)         /* Now fix MOD entries.   */
          DISTLIB(AOS12)   /* MOD DLIB.              */
          FMID(FXY1102)    /* Functional owner.      */
          LMOD(LMOD01)     /* Connect to LMOD.       */
          /*                          */
ENDUCL                                /*                          */

```

**Note:** In this example, the entire set of control cards needed to link the load module was specified. MOD entries for modules within the load module were also updated.

### Example 2: Changing the Link-Edit Attributes of an LMOD

As in the previous example, the correct way to change any part of a LMOD entry is through the use of JCLIN. However, at times, assuming a knowledge of SMP/E and UCL processing, you may decide to use UCL to do it. For this example, assume you have a load module, LMOD99, with link-edit attributes of RENT REUS, and that you have made changes so that it is no longer reentrant but is now authorized. This change can be made as follows:

```

SET      BDY(TGT1)          /* Set to target zone.    */
UCLIN                                /*                          */
DEL      LMOD(LMOD99)       /* Existing LMOD entry.   */
          RENT.             /* Delete RENT attribute. */
ADD      LMOD(LMOD99)       /* Existing LMOD entry.   */
          AC=1              /* Add authorization.     */
ENDUCL                                /*                          */

```

### Example 3: Deleting a MODEL Subentry

Suppose load module LMOD03 for product A included module MOD01 from product B, and both products are installed in zone MVS1. Product B deleted MOD01 without replacing it, and as a result, SMP/E created a MODEL subentry for MOD01 in the LMOD entry for LMOD03. You have decided to reintroduce your own version of MOD01, but do not want it relinked into LMOD03. To prevent this relinking, delete the MODEL subentry from the LMOD entry, as shown below:

```
SET      BDY(MVS1)           /* Set to MVS zone.          */
UCLIN                               /* Start UCLIN processing.   */
DEL      LMOD(LMOD03)        /* Identify LMOD entry.      */
          MODEL(MOD01)       /* Delete MODEL subentry.    */
ENDUCL                               /*                            */
```

### Example 4: Completing Cross-Zone Updates

If SMP/E could not complete cross-zone updates for APPLY or RESTORE processing, you may need to use UCLIN to make the remaining changes. (SMP/E issues messages and reports to tell you what cross-zone work you may need to finish.) Here are some examples of using UCLIN to change XZMOD subentries in LMOD entries. Make sure you check the messages and reports to determine whether any additional changes are needed for cross-zone subentries in MOD entries or TARGETZONE entries.

- **Adding XZMOD subentries:** This example adds module IGCXXX from zone IMS1 and module IGCABC from zone NCP1 to the IEANUC01 LMOD entry. If this LMOD entry previously had no XZMOD subentries, SMP/E automatically sets XZMODP to indicate that the LMOD entry now contains XZMOD subentries.

```
SET BDY(MVS1)           /* set to MVS zone          */
UCLIN                   /* start UCLIN processing   */
ADD LMOD(IEANUC01)      /* identify LMOD entry      */
      XZMOD((IGCXXX,IMS1), /* add module from IMS1 zone */
            (IGCABC,NCP1)) /* add module from NCP1 zone */
ENDUCL                  /* end UCLIN processing     */
```

- **Replacing XZMOD subentries:** This example replaces all the XZMOD subentries in LMOD entry CICSDIAG with a single subentry for module ISPLINK from zone MVS1. If this LMOD entry previously had no XZMOD subentries, SMP/E automatically sets XZMODP to indicate that the LMOD entry now contains XZMOD subentries.

```
SET BDY(CICS1)          /* set to CICS zone         */
UCLIN                   /* start UCLIN processing   */
REP LMOD(CICSDIAG)      /* identify LMOD entry      */
      XZMOD((ISPLINK,MVS1)) /* replace list with one value */
ENDUCL                  /* end UCLIN processing     */
```

- **Deleting XZMOD subentries:** This example shows how to delete a single XZMOD subentry or all XZMOD subentries.
  - For LMOD entry IMSDIAG1, one XZMOD subentry is deleted. If no XZMOD values are left, SMP/E turns the XZMODP indicator off to indicate that the LMOD entry no longer has XZMOD subentries.
  - For LMOD entry IMSDIAG2, all XZMOD subentries are deleted. SMP/E turns the XZMODP indicator off to indicate that the LMOD entry no longer contains XZMOD subentries.

```

SET BDY(IMS1)           /* set to IMS zone           */.
UCLIN                  /* start UCLIN processing    */.
DEL LMOD(IMS1)         /* identify LMOD entry       */.
    XZMOD((ISPLINK,MVS1)) /* delete ISPLINK reference  */.
DEL LMOD(IMS2)         /* identify LMOD entry       */.
    XZMOD()            /* delete entire list        */.
ENDUCL                /* end UCLIN processing      */.
    
```

### Example 5: Adding a CALLLIBS Subentry List to an LMOD Entry

You can use either the ADD or the REP statement to add a CALLLIBS subentry list to an LMOD entry, depending on whether the CALLLIBS already exists or not. The order in which the libraries are specified is important because it indicates the order in which the SYSLIB concatenation is built. This is shown in the examples that follow.

- **Adding a CALLLIBS subentry:** This example adds a CALLLIBS subentry list containing PLIBASE and APPLIB to LMOD entry LMOD04.

```

SET BDY(MVS1)           /* set to MVS zone           */.
UCLIN                  /* start UCLIN processing    */.
ADD LMOD(LMOD04)       /* identify LMOD entry       */.
    CALLLIBS(PLIBASE,APPLIB) /* add CALLLIBS subentry    */.
ENDUCL                /* end UCLIN processing      */.
    
```

- **Adding to an existing CALLLIBS subentry:** Suppose LMOD entry LMOD05 has a CALLLIBS subentry list containing PLIBASE and APPLIB, and CSSLIB is to be added to this list. The entire CALLLIBS subentry must be replaced as shown below:

```

SET BDY(MVS1)           /* set to MVS zone           */.
UCLIN                  /* start UCLIN processing    */.
REP LMOD(LMOD05)       /* identify LMOD entry       */.
    CALLLIBS(PLIBASE,APPLIB, /* /*replace entire CALLLIBS */.
             CSSLIB)
ENDUCL.                /* end UCLIN processing      */.
    
```

**Note:** If an ADD statement is used to try to update an existing CALLLIBS subentry list in an LMOD entry, SMP/E issues an error message.

### Example 6: Deleting Link-Edit Control Statements

Suppose you have installed a SYSMOD that was packaged incorrectly and added unwanted link-edit control statements to the LMOD entry for an existing load module. As a result, you now need to delete those statements from the LMOD

## LMOD Entry (Distribution and Target Zone)

---

entry. Here is an example of the UCLIN statements you can use to make this change:

```
SET BDY(MVS1)          /* set to MVS zone          */.  
UCLIN                 /* start UCLIN processing  */.  
DEL LMOD(LMODXZ)      /* identify LMOD entry     */.  
++LMODIN              /* delete all link-edit    */.  
++ENDLMODIN          /* control statements.     */.  
ENDUCL                /* end UCLIN processing    */.
```

Using the REP command instead of the DEL command gives you the same result. There is no need to specify the control statements to be deleted. In fact, you cannot selectively delete link-edit control statements from an LMOD entry—you can only delete them all.



---

## MAC Entry (Distribution and Target Zone)

The MAC entry describes a macro that exists in the distribution or target macro libraries. A MAC entry is generally created by one of the following methods:

- **Installing a SYSMOD that contains the macro.** MAC entries are created the first time you install a SYSMOD containing a ++MAC statement for a macro that does not yet have a MAC entry.
- **Processing JCLIN.** MAC entries can be built during JCLIN processing when SMP/E scans the assembler statements for inline assemblies to determine which macros are used in that assembly. The name of the assembly is kept in the MAC entry, so when the macro is updated, SMP/E can reassemble the required modules.

MAC entries can also be built when SMP/E scans copy steps and finds a SELECT statement that specifies TYPE=MAC.

For additional information, see “Processing” on page 179.

SMP/E records the function and service level of each macro in the MAC entry, as well as information about how that macro affects the structure of the distribution or target libraries and modules.

Once a MAC entry exists for a macro, it is updated as subsequent SYSMODs are installed.

## Subentries

These are the subentries for the MAC entry as they appear in the LIST output:

*name*

is the name of the macro represented by the MAC entry.

The name can contain from 1 to 8 alphanumeric characters and \$, #, @, or hex C0.

### **DISTLIB**

is the ddname of the macro distribution library.

The UCL operand is **DISTLIB**(*ddname*).

- The ddname can contain from 1 to 8 alphanumeric characters.
- The DISTLIB subentry is not required when the MAC entry is first defined. For example, during JCLIN processing SMP/E builds MAC entries without a DISTLIB value. However, SMP/E can add a DISTLIB value later when it processes the ++MAC statement. A DISTLIB value is needed at some time to process any changes for the macro.

### **FMID**

identifies the functional owner of this macro. The functional owner is the last function SYSMOD that replaced this macro.

The UCL operand is **FMID**(*sysmod\_id*).

The SYSMOD ID must contain 7 alphanumeric characters.

### GENASM

identifies those assemblies that have to be done during APPLY each time this macro is modified. These assemblies must exist as either ASSEM or SRC entries in the target zone.

The UCL operand is **GENASM**(*name...*).

The names can contain from 1 to 8 alphanumeric characters.

### LASTUPD

identifies the cause of the last change to this MAC entry.

The UCL operand is **LASTUPD**(*value*). This subentry can contain one of the following values:

#### JCLIN

indicates that the change was made during JCLIN command processing.

#### UCLIN

indicates that the change was made as a result of UCLIN processing.

#### sysmod-id

indicates that the change was made during the installation of the indicated SYSMOD.

The SYSMOD ID must contain 7 alphanumeric characters.

### LASTUPD TYPE

indicates how the entry was last changed.

The UCL operand is **LASTUPDTYPE**(*value*). This subentry can contain one of the following values:

<b>ADD</b>	The entry was added.
<b>MOV</b>	The entry was moved.
<b>UPD</b>	The entry was updated.

### MALIAS

identifies any alias names for this macro.

The UCL operand is **MALIAS**(*name...*).

The alias names can contain from 1 to 8 alphanumeric characters.

### RMID

identifies the last SYSMOD that **replaced** this macro. Any subsequent SYSMOD that modifies this macro must have a defined relationship (such as PRE or SUP) with this SYSMOD.

The UCL operand is **RMID**(*sysmod\_id*).

- The SYSMOD ID must contain 7 alphanumeric characters.
- If RMID is not specified but FMID is, SMP/E sets the RMID value to the specified FMID.

### SYSLIB

is the ddname of the target system macro library.

The UCL operand is **SYSLIB**(*ddname*).

- Only one SYSLIB value may be specified.
- The ddname can contain from 1 to 8 alphanumeric characters.



**UMID**

identifies all those SYSMODs that have **updated** this macro since it was last replaced. Any subsequent SYSMOD that modifies this macro must have a defined relationship (such as PRE or SUP) with all these SYSMODs.

The UCL operand is **UMID(sysmod\_id...)**.

The SYSMOD ID must contain 7 alphanumeric characters.

**LIST Examples**

To list all the MAC entries in a particular zone, you can use the following commands:

```
SET    BDY(TGT1)           /* Set to requested zone. */.
LIST   MAC                 /* List all MAC entries. */.
```

To list specific MAC entries, you can use these commands:

```
SET    BDY(TGT1)           /* Set to requested zone. */.
LIST   MAC(MAC01          /* List only these two */
        MAC02)            /* entries. */.
```

The format of the LIST output for each MAC entry is the same for both of these commands. The only difference is the number of MAC entries listed. Figure 115 shows an example of LIST output for MAC entries.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
TGT1      MACRO ENTRIES

NAME

MAC01     LASTUPD          = JXY1102 TYPE=ADD
          LIBRARIES       = DISTLIB=DLIBMAC1 SYSLIB=MACLIB01
          FMID             = JXY1102
          RMID             = JXY1102
          MALIAS           = TERMINAL TERM T
          GENASM           = ASSEM01 ASSEM02 SRC01 SRC02

MAC02     LASTUPD          = JXY1000 TYPE=UPD
          LIBRARIES       = DISTLIB=DLIBMAC1 SYSLIB=MACLIB01
          FMID             = JXY1121
          RMID             = UZ00010
          UMID             = UZ00014 UZ00015
          GENASM           = ASSEM01 SRC02
```

Figure 115. MAC Entry: Sample LIST Output

By specifying the FORFMID operand, you can reduce the number of MAC entries listed. When FORFMID is specified, SMP/E lists a MAC entry only if its FMID matches one of the FMIDs specified on the FORFMID operand. For example, to list MAC entries whose FMIDs either are defined in FMIDSET TP or are JXY1102, you can use these commands:

```
SET    BDY(TGT1)           /* Set to target zone. */.
LIST   MAC                 /* List all macro entries */
        FORFMID(TP        /* for the TP FMIDSET */
                JXY1102) /* and FMID JXY1102. */.
```

You can use the LIST command to find out the names of all SYSMODs that have modified a macro. To include the names of these SYSMODs in the LIST output, you can use the XREF operand, as shown in these commands:

```
SET      BDY(TGT1)          /* Set to requested zone.  */.  
LIST     MAC                /* List all macro entries  */.  
         XREF               /* and related SYSMODs.   */.
```

### Notes:

1. XREF can be used either in mass mode or in select mode.
2. SMP/E obtains the data included for the XREF operand by checking for MAC and MACUPD entries for this macro in all the SYSMOD entries. Because this data is not contained in the MAC entry itself, you cannot use UCLIN to change it in the MAC entry.

Figure 116 is an example of the LIST output produced when the XREF operand is used.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT  
TGT1      MACRO ENTRIES  
  
NAME  
MAC01     LASTUPD          = JXY1102 TYPE=ADD  
          LIBRARIES        = DISTLIB=DLIBMAC1 SYSLIB=MACLIB01  
          FMID              = JXY1102  
          RMID              = JXY1102  
          MALIAS            = TERMINAL TERM      T  
          GENASM            = ASSEM01 ASSEM02 SRC01 SRC02  
          SYSMOD HISTORY    = SYSMOD TYPE      DATE MCS  --STATUS--  
                           JXY1102 FUNCTION 90.100 MAC  APP  ACC  
  
MAC02     LASTUPD          = JXY1000 TYPE=UPD  
          LIBRARIES        = DISTLIB=DLIBMAC1 SYSLIB=MACLIB01  
          FMID              = JXY1121  
          RMID              = UZ00010  
          UMID              = UZ00014 UZ00015  
          GENASM            = ASSEM01 SRC02  
          SYSMOD HISTORY    = SYSMOD TYPE      DATE MCS  --STATUS--  
                           JXY1102 FUNCTION 90.100 MAC  APP  ACC  
                           JXY1121 FUNCTION 90.150 MAC  APP  ACC  
                           UZ00010 PTF      90.150 MAC  APP  
                           UZ00014 PTF      90.160 MACUPD APP  
                           UZ00015 PTF      90.161 MACUPD APP
```

Figure 116. MAC Entry: Sample LIST Output When XREF Is Specified

## UNLOAD Examples

To dump the MAC entries in UCL format, you can use the UNLOAD command. To unload all the MAC entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested zone.  */.  
UNLOAD   MAC                /* Unload all MAC entries. */.
```

To unload specific MAC entries, you can use these commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD  MAC(MAC01          /* Unload only these two */.
        MAC02).           /* entries.                */.
```

The format of the UNLOAD output for each MAC entry is the same for both of these commands. The only difference is the number of MAC entries listed. Figure 117 is an example of UNLOAD output for MAC entries.

```
UCLIN .
REP     MAC                ( MAC01   )
        LASTUPD           ( JXY1102 )
        LASTUPDTYPE      ( ADD   )
        DISTLIB           ( DLIBM01 )
        SYSLIB            ( MACLIB01 )
        FMID              ( JXY1102 )
        RMID              ( JXY1102 )
        MALIAS            ( TERMINAL TERM   T   )
        GENASM            ( ASSEM01 ASSEM02 SRC01 SRC02 )
.
REP     MAC                ( MAC02   )
        LASTUPD           ( JXY1121 )
        LASTUPDTYPE      ( UPD   )
        DISTLIB           ( DLIBM01 )
        SYSLIB            ( MACLIB01 )
        FMID              ( JXY1121 )
        RMID              ( UZ00010 )
        UMID              ( UZ00014 UZ00015 )
        GENASM            ( ASSEM01 SRC02 )
.
ENDUCL.
```

Figure 117. MAC Entry: Sample UNLOAD Output

By specifying the FORFMID operand, you can reduce the number of MAC entries unloaded. When FORFMID is specified, SMP/E unloads a MAC entry only if its FMID matches one of the FMIDs specified on the FORFMID operand. For example, to unload MAC entries whose FMIDs either are defined in FMIDSET TP or are JXY1102, you can use these commands:

```
SET      BDY(TGT1)          /* Set to target zone. */.
UNLOAD  MAC                /* Unload all macro entries */.
        FORFMID(TP        /* for the TP FMIDSET */.
        JXY1102)         /* and FMID JXY1102.    */.
```

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the MAC entry. After the UCLIN changes are made, the MAC entry must contain at least the following subentries:

- FMID
- RMID

Otherwise, there is not enough information in the entry to process the macro. If any of these subentries are missing, SMP/E does not make the requested UCL updates to the entry, and the entry remains as it was before the UCL command.

The following examples are provided to help you use the MAC entry:

- “Example 1: Adding a New MAC Entry”
- “Example 2: Defining an Alias for an Existing Macro”

### Example 1: Adding a New MAC Entry

Defining a new macro entry with UCL is very seldom required; generally, MAC entries are created from the information specified on the ++MAC statements contained in the SYSMODs when SYSMODs are installed. If, however, you want to use UCL in defining a new macro entry, the following is an example of the minimum information you should provide:

```

SET      BDY(TGT1)          /* Set to target zone.      */.
UCLIN                               /*                          */.
ADD      MAC(MAC01)         /* Define new macro entry.  */.
          DISTLIB(AMACLIB)  /* Define DLIB,             */.
          SYSLIB(MACLIB)    /* system library          */.
                               /* (never the SMPMTS or MTS). */.
          FMID(ZUSR001)     /* Functional owner (in this */.
                               /* example a user function). */.
                               /*                          */.
ENDUCL                               /*                          */.
SET      BDY(DLB1).         /* Now do same to DLIB.    */.
UCLIN                               /*                          */.
ADD      MAC(MAC01)         /* Define new macro entry.  */.
          DISTLIB(AMACLIB)  /* Define DLIB.            */.
                               /* No SYSLIB info in DLIB.  */.
          FMID(ZUSR001)     /* Functional owner (in this */.
                               /* example a user function). */.
                               /*                          */.
ENDUCL                               /*                          */.

```

### Example 2: Defining an Alias for an Existing Macro

The following defines the method of adding an alias to an existing macro:

```

SET      BDY(TGT1)          /* Set to target zone.      */.
UCLIN                               /*                          */.
ADD      MAC(MAC01)         /* Existing macro entry.    */.
          MALIAS(MAC01AL1)  /* New alias name.         */.
                               /* End of adding macro.    */.
ENDUCL                               /*                          */.
SET      BDY(DLB1)         /* Now do same thing to    */.
                               /* appropriate DLIB.       */.
UCLIN                               /*                          */.
ADD      MAC(MAC01)         /* Existing macro entry.    */.
          MALIAS(MAC01AL1)  /* New alias name.         */.
                               /* End of adding macro.    */.
ENDUCL                               /*                          */.

```

**Note:** This UCL does not create an alias entry in the target or distribution libraries; that must be done outside of SMP/E using standard utilities. This ensures that, when the macro is subsequently modified, both the major entry and the alias entry are updated.

## MCS Entry (SMPPTS)

The MCS entry is a copy of a SYSMOD exactly as it was received from the SMPPTFIN data set. The MCS entry is in the SMPPTS data set, which is used as a warehouse for SYSMODs. When SMP/E receives a SYSMOD, it stores the SYSMOD as a separate member in the SMPPTS. The member name matches the SYSMOD ID, and each member is an MCS entry. SMP/E also creates a SYSMOD entry in the global zone to describe the SYSMOD that was received. Thus, the MCS entry and the global zone SYSMOD entry are closely related.

When SMP/E accepts or applies SYSMODs, it gets them from the MCS entries in the SMPPTS. An MCS entry is generally kept in the SMPPTS until the associated SYSMOD is accepted; then the entry is deleted.

- You may want SMP/E to save the MCS entries after ACCEPT processing, for example, if you plan to do a system generation. To do this, specify **NOPURGE** in the OPTIONS entry that is in effect during ACCEPT processing.
- Likewise, you may want to save the MCS entries after RESTORE processing. To do this, specify **NOREJECT** in the OPTIONS entry that is in effect during RESTORE processing.

## Subentries

The MCS entry contains no SMP/E data and appears to the system as a member of a normal partitioned data set.

## LIST Examples

To list all the MCS entries in the SMPPTS, you can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to global.      */
LIST     MCS               /* List all MCS entries. */
```

To list specific MCS entries, you can use these commands:

```
SET      BDY(GLOBAL)      /* Set to global.      */
LIST     MCS(UZ12345,     /* List only these two  */
          UZ12346)        /* entries.             */
```

The format of the LIST output for each MCS entry is the same for both of these commands. The only difference is the number of MCS entries listed.

Figure 118 on page 722 is an example of LIST output for MCS entries.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
```

```
SMPPTS      MCS ENTRIES
```

```
NAME
```

```
UZ12345  MCS      = ++PTF(UZ12345).
                ++VER(Z038) FMID(JXY1102).
                ++MOD(XYMOD01) DISTLIB(AOS12).

UZ12346  MCS      = ++PTF(UZ12346).
                ++VER(Z038) FMID(FXY1102).
                ++JCLIN.
                //JOB      JOB 'accounting info',MSGLEVEL=(1,1)
                //STEP1    EXEC PGM=IEBCOPY
                //MACLIB   DD DSN=SYS1.MACLIB,DISP=SHR
                //AMACLIB  DD DSN=SYS1.AMACLIB,DISP=SHR
                //SYSIN   DD *
                COPY INDD=AMACLIB,OUTDD=MACLIB
                /*
                ++MAC(MAC01) DISTLIB(AMACLIB).
```

Figure 118. MCS Entry: Sample LIST Output

As the example for UZ12346 shows, SMP/E includes inline JCLIN when listing the MCS entries.

You can use various SYSMOD-related LIST operands to limit which MCS entries are listed. For more information, see “SYSMOD Entry (Distribution and Target Zone)” on page 759 and “SYSMOD Entry (Global Zone)” on page 774.

In addition to the LIST command, you can use standard system utility programs (such as IEBGENER, IEBPTPCH, IEHLIST, and so on) or products such as ISPF to display these entries or information about the data set.

## UCLIN Examples

You cannot use UCLIN to add, update, or delete MCS entries. However, you can use the REJECT command to delete MCS entries. For more information, see Chapter 13.

**Do not use** system utility programs to update MCS entries. The global zone SYSMOD entry is coordinated with the MCS entry. Any changes to the MCS entry made outside of SMP/E may get these entries out of synchronization and can result in unpredictable results when the associated SYSMOD is processed.

---

## MOD Entry (Distribution and Target Zone)

The MOD entry describes a particular module, its function and service level, and how it relates to a load module in the target library. A MOD entry is generally created by one of the following methods:

- **Installing a SYSMOD that contains the module.** MOD entries are created the first time you install a SYSMOD that contains a ++MOD statement for a module that does not yet have a MOD entry. If the module comes from a copied distribution library, SMP/E also builds an LMOD entry with the same name. For additional information about copied libraries and the creation of LMOD entries, see “Processing” on page 77, “Usage Notes” on page 67, and “DLIB Entry (Distribution and Target Zone)” on page 670.

A MOD entry can also be created when you install a SYSMOD that causes the assembled source to be linked to the distribution library. This can happen when a SYSMOD contains both a ++MOD and a ++SRC statement for the same module, or when the DISTMOD operand is specified on the ++SRC statement.

- **Processing JCLIN.** MOD entries can be built during JCLIN processing when SMP/E scans the link-edit and copy steps. A MOD entry built during JCLIN processing is always associated with an LMOD entry. Thus, when the module is changed, SMP/E can determine which load modules are affected. For additional information, see “Processing” on page 179.

## Subentries

These are the subentries for the MOD entry as they appear in the LIST output:

*name*

is the name of the module represented by the MOD entry.

The name can contain from 1 to 8 alphanumeric characters and \$, #, @, or hex C0.

### ASSEMBLE

indicates that the source for this module must always be assembled, even if the object module is supplied in the SYSMOD.

The UCL operand is **ASSEMBLE**.

### CSECT

specifies the CSECTs present in this module.

The UCL operand is **CSECT(name...)**.

- The CSECT subentry is not required. However, if CSECT is missing, SMP/E assumes that the module contains only one CSECT, whose name matches the module name.
- A CSECT name can contain from 1 to 8 characters.

The name can contain any characters except the following:

Comma ,  
 Left parenthesis (  
 Right parenthesis )  
 Blank

- Comments are not allowed within a CSECT name. For example, the following is not allowed:

```
CSECT ( /* this is a csect name */ CSECT01)
```

The comment is interpreted as part of the CSECT name, instead of a comment.

- The list of CSECT names must include all of the CSECTs in the module, even if one of the CSECTs matches the module name.
- Once a ++MOD with the CSECT operand is processed, SMP/E saves the CSECT information in the MOD entry. From then on, SMP/E uses that saved information if it is not supplied on subsequent SYSMODs.
- If a module is changed from multiple CSECTs to a single CSECT that matches the module name, the CSECT operand must be specified with one name in order to get SMP/E to store that information in the MOD entry.

### DALIAS

specifies the alias name for the module, where the alias exists only in the distribution library.

The UCL operand is **DALIAS**(*name*).

The DALIAS name can contain from 1 to 8 alphanumeric characters.

### DISTLIB

is the ddname of the module distribution library.

The UCL operand is **DISTLIB**(*ddname*).

The ddname can contain from 1 to 8 alphanumeric characters.

### FMID

identifies the functional owner of this module. The functional owner is the last function SYSMOD that replaced this module.

The UCL operand is **FMID**(*sysmod\_id*).

- The SYSMOD ID must contain 7 alphanumeric characters.
- Some MOD entries, specifically those associated with a system generation assembly, may have no functional owner. The DISTLIB for these modules is SYSPUNCH. They may have either no FMID, or one of the following values:
  - If the module has no functional owner but was assembled during system generation, \***SYSGEN** appears as the FMID.
  - If the module has no functional owner and was not assembled during system generation, ??????? appears as the FMID.

### LASTUPD

identifies the cause of the last change to this MOD entry.

**Note:** If a given UCLIN command specifies only cross-zone subentries, this field is not changed.



The UCL operand is **LASTUPD**(*value*). This subentry can contain one of the following values:

**JCLIN**

indicates that the change was made during JCLIN command processing.

**UCLIN**

indicates that the change was made as a result of UCLIN processing.

**sysmod-id**

indicates that the change was made during installation of the specified SYSMOD. The SYSMOD did one of the following:

- Contained inline JCLIN that affected the module
- Changed the distribution library for the module through the DISTLIB operand on the ++MOD statement
- Added the module to an existing load module through the LMOD operand on the ++MOD statement

The SYSMOD ID must contain 7 alphanumeric characters.

**LASTUPD TYPE**

indicates how the entry was last changed.

**Note:** If a given UCLIN command specifies only cross-zone subentries, this field is not changed.

The UCL operand is **LASTUPDTYPE**(*value*). This subentry can contain one of the following values:

<b>ADD</b>	The entry was added.
<b>DEL</b>	A subentry in the entry was deleted.
<b>MOV</b>	The module was moved.
<b>UPD</b>	The entry was updated.

**LKED ATTRIBUTES**

identifies the link-edit attributes that must be used when this module is link-edited. SMP/E supports the following link-edit attributes. For more information, see your link-edit utility reference manual.

**AC=1**

specifies that the AC=1 parameter, which is the authorization code, is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **AC=1**.

**ALIGN2**

specifies that the ALIGN2 parameter (alignment on a 2KB boundary) is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **ALIGN2** or **ALN2**.

**AMODE=24**

specifies that the AMODE=24 parameter is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **AMODE=24** or **AMOD=24**.

### **AMODE=31**

specifies that the **AMODE=31** parameter is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **AMODE=31** or **AMOD=31**.

### **AMODE=ANY**

specifies that the **AMODE=ANY** parameter is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **AMODE=ANY** or **AMOD=ANY**.

### **AMODE=MIN**

specifies that the **AMODE=MIN** parameter is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **AMODE=MIN** or **AMOD=MIN**.

### **DC**

specifies that the **DC** parameter, which is the downward compatible attribute, is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **DC**.

### **FETCHOPT**

specifies that the **FETCHOPT** parameter is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **FETCHOPT(PACK | NOPACK, PRIME | NOPRIME)**.

### **MAXBLK**

specifies that the **MAXBLK** parameter is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **MAXBLK(nnnnn)**, where *nnnnn* is a number between 256 and 32760.

### **NE**

specifies that the **NE** parameter, which is the noneditable attribute, is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **NE**.

### **NOCALL**

specifies that the **NOCALL** parameter is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **NOCALL** or **NCAL**.

### **OL**

specifies that the **OL** parameter is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **OL**.

### **OVLV**

specifies that the **OVLV** parameter, which specifies that the module is in overlay structure, is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **OVLV**.

**REFR**

specifies that the REFR parameter, which is the refreshable attribute, is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **REFR**.

**RENT**

specifies that the RENT parameter, which indicates that the module is reentrant, is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **RENT**.

**REUS**

specifies that the REUS parameter, which indicates that the module is reusable, is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **REUS**.

**RMODE=24**

specifies that the RMODE=24 parameter is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **RMODE=24** or **RMOD=24**.

**RMODE=ANY**

specifies that the RMODE=ANY parameter is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **RMODE=ANY** or **RMOD=ANY**.

**SCTR**

specifies that the SCTR parameter, which indicates that the module can be scatter-loaded, is to be passed to the link-edit utility when the module is link-edited.

The UCL operand is **SCTR**.

**STD**

is a special SMP/E indication that the module should be link-edited with none of the above attributes.

When this indicator is present and a link-edit is to be done, SMP/E will pass the link-edit utility only those parameters specified in the appropriate link-edit UTILITY entry.

The UCL operand is **STD**.

**LMOD**

specifies the names of the load modules into which this module was copied or link-edited on the target system.

The UCL operand is **LMOD**(*name...*).

- The load module names can contain from 1 to 8 alphanumeric characters.
- If there are no LMOD names in a MOD entry, SMP/E assumes that the module was not selected when the SYSMOD containing the module was initially installed. Therefore, during APPLY processing, SMP/E will not link or copy the module into any target library.

### RMID

identifies the last SYSMOD that **replaced** this module. Any subsequent SYSMOD that modifies this module must have a defined relationship (such as PRE or SUP) with this SYSMOD.

The UCL operand is **RMID**(*sysmod\_id*).

- The SYSMOD ID must contain 7 alphanumeric characters.
- If **RMID** is not specified but **FMID** is, SMP/E sets the RMID subentry to the FMID value.
- RMID is not required for a module assembled during system generation. The DISTLIB for these modules is SYSPUNCH.

### RMIDASM

specifies that the last replacement (RMID) to the module was done by a SYSMOD which caused an assembly of the module as a result of a source or macro modification.

The UCL operand is **RMIDASM**.

### TALIAS

specifies one or more alias names for the module, where the alias exists in the distribution library and, for copied modules, also in the target library.

The UCL operand is **TALIAS**(*name...*).

The alias names can contain from 1 to 8 alphanumeric characters.

### UMID

identifies all those SYSMODs that have **updated** this module since it was last replaced. Any subsequent SYSMOD that modifies this module must have a defined relation (such as PRE or SUP) with all these SYSMODs.

The UCL operand is **UMID**(*sysmod\_id...*).

The SYSMOD ID must contain 7 alphanumeric characters.

### XZLMOD

specifies one or more load modules in other zones into which this module was added by the LINK command. The name of the zone containing each load module is also indicated. Any zone updated with the LINK command or cross-zone information **cannot** be processed by SMP/E Release 6 or earlier.

The UCL operand is **XZLMOD**((*load module,zone*)...).

- The zone name specified for an XZLMOD subentry cannot match the name of the set-to zone.
- An entry can contain XZLMOD and XZLMODP subentries without any other subentries.
- If UCLIN is used to update an existing MOD entry and **only** cross-zone subentries are changed (XZLMOD and XZLMODP), SMP/E does not update the LASTUPD and LASTUPDTYPE subentries.
- The XZLMOD subentry is added to a MOD entry automatically during LINK command processing. However, it is **never** automatically removed.

**XZLMODP**

indicates that this module has been linked into one or more load modules controlled by a different target zone, and that XZLMOD subentries exist in this MOD entry.

The UCL operand is **XZLMODP**.

- It is never necessary to specify the XZLMODP subentry on a UCL statement. SMP/E automatically determines the setting of XZLMODP, according to whether the MOD entry contains XZLMOD subentries.
- You cannot add the XZLMODP subentry to a MOD entry that does not contain XZLMOD subentries.
- You cannot delete the XZLMODP subentry from a MOD entry containing XZLMOD subentries.
- An entry can contain XZLMOD and XZLMODP subentries without any other subentries.
- If UCLIN is used to update an existing MOD entry and **only** cross-zone subentries are changed (XZLMOD and XZLMODP), SMP/E does not update the LASTUPD and LASTUPDTYPE subentries.

**LIST Examples**

To list all the MOD entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)           /* Set to requested zone. */.
LIST     MOD                 /* List all MOD entries.  */.
```

To list specific MOD entries, you can use these commands:

```
SET      BDY(TGT1)           /* Set to requested zone. */.
LIST     MOD(MOD01           /* List only these two   */.
          MOD02)            /* entries.              */.
```

The format of the LIST output for each MOD entry is the same for both of these commands. The only difference is the number of MOD entries listed. The following examples show LIST output for MOD entries. Figure 119 on page 730 does not have cross-zone subentries. Figure 120 on page 731 does.

## MOD Entry (Distribution and Target Zone)

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT

TGT1      MODULE ENTRIES

NAME

ASSEM01  LASTUPD      = JCLIN  TYPE=ADD
          LIBRARIES   = DISTLIB=SYSPUNCH
          LMOD        = LMOD99A  LMOD99B

ASSEM02  LASTUPD      = JCLIN  TYPE=ADD
          LIBRARIES   = DISTLIB=SYSPUNCH
          LMOD        = ASSEM02

MOD01    LASTUPD      = JXY1102 TYPE=ADD
          LIBRARIES   = DISTLIB=AOS12
          FMID       = JXY1102
          RMID       = JXY1102
          CSECT      = MOD01C1  MOD01C2  MOD01C3
          LMOD       = LMOD01

MOD02    LASTUPD      = JXY1102 TYPE=ADD
          LIBRARIES   = DISTLIB=AOS12
          FMID       = JXY1102
          RMID       = JXY1102
          CSECT      = MOD02C1
          LMOD       = LMOD01  LMOD02

MOD03    LASTUPD      = JXY1121 TYPE=UPD
          LIBRARIES   = DISTLIB=AOS12
          FMID       = JXY1121
          RMID       = JXY1121
          LKED ATTRIBUTES = RENT    REUS    AC=1
          UMID       = UZ00010  UZ00014
          LMOD       = LMOD03

MOD04    LASTUPD      = JXY1121 TYPE=UPD
          LIBRARIES   = DISTLIB=AOS12
          ASSEMBLE
          FMID       = JXY1121
          RMID       = JXY1121
                   RMIDASM
          LMOD       = MOD04

MOD05    LASTUPD      = JXY0001 TYPE=ADD
          LIBRARIES   = DISTLIB=DLIB1
          FMID       = JXY0001
          RMID       = JXY0001
          LKED ATTRIBUTES = NCAL    OL    AMODE=MIN

MOD06    LASTUPD      = JXY0001 TYPE=ADD
          LIBRARIES   = DISTLIB=DLIB1
          FMID       = JXY0001
          RMID       = JXY0001
          LKED ATTRIBUTES = FETCHOPT(PACK,PRIME)  AMODE=MIN
```

Figure 119. MOD Entry: Sample LIST Output (No Cross-Zone Subentries)

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMP LIST OUTPUT

NOW SET TO TARGET ZONE CICS1

CICS1      MODULE ENTRIES

NAME

CICSMOD1  LASTUPD      = LINK    TYPE=UPD
           LIBRARIES   = DISTLIB=ARESLIB
           FMID        = HCI1703
           RMID        = HCI1703
           LMOD        = CICSMOD1
           XZLMODP     =
           XZLMOD      = LMOD01  IN ZONE MVS1
                       LMOD02  IN ZONE MVS1
                       LMOD01  IN ZONE MVS2

CICSMOD2  LASTUPD      = LINK    TYPE=UPD
           LIBRARIES   = DISTLIB=ARESLIB
           FMID        = HCI1703
           RMID        = HCI1703
           LMOD        = CICSMOD2
           XZLMODP     =
           XZLMOD      = LMOD01  IN ZONE MVS1
                       LMOD02  IN ZONE MVS1
                       LMODAA01 IN ZONE MVS2
                       LMOD01  IN ZONE MVS2

CICSMOD3  LASTUPD      = HCI1703 TYPE=ADD
           LIBRARIES   = DISTLIB=ARESLIB
           FMID        = HCI1703
           RMID        = HCI1703
           LMOD        = CICS LMD2

CICSMOD4  XZLMODP     =
           XZLMOD      = LMOD01  IN ZONE MVS1
                       LMOD01  IN ZONE MVS2
    
```

Figure 120. MOD Entry: Sample LIST Output (Cross-Zone Entries)

By specifying the FORFMID operand, you can reduce the number of MOD entries listed. When **FORFMID** is specified, SMP/E lists a MOD entry only if its FMID matches one of the FMIDs specified on the FORFMID operand. For example, to list MOD entries whose FMIDs are either defined in FMIDSET TP or are JXY1102, you can use these commands:

```

SET      BDY(TGT1)          /* Set to target zone.    */
LIST     MOD                /* List all MOD entries   */
          FORFMID(TP        /* for the TP FMIDSET     */
                JXY1102)   /* and FMID JXY1102.     */
    
```

You can also use the LIST command to find out the names of all SYSMODs that have updated a module. To include the names of these SYSMODs in the LIST output, you can use the XREF operand, as shown in these commands:

```

SET      BDY(TGT1)          /* Set to requested zone. */
LIST     MOD                /* List all module entries */
          XREF              /* and related SYSMODs.   */
    
```

## MOD Entry (Distribution and Target Zone)

### Notes:

1. XREF can be used either in mass mode or in select mode.
2. SMP/E obtains the data included for the XREF operand by checking all the SYSMOD entries to find which SYSMODs contained ++MOD or ++ZAP statements for this module. Because this data is not contained in the MOD entry itself, you cannot use UCLIN to change it in the MOD entry.

Figure 121 is an example of the LIST output produced when the XREF operand is used.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMLIST OUTPUT
TGT1      MODULE ENTRIES

NAME
ASSEM01  LASTUPD      = JCLIN  TYPE=ADD
          LIBRARIES   = DISTLIB=SYSPUNCH
          LMOD        = LMOD99A  LMOD99B

ASSEM02  LASTUPD      = JCLIN  TYPE=ADD
          LIBRARIES   = DISTLIB=SYSPUNCH
          LMOD        = ASSEM02

MOD01    LASTUPD      = JXY1102 TYPE=ADD
          LIBRARIES   = DISTLIB=AOS12
          FMID        = JXY1102
          RMID        = JXY1102
          CSECT       = MOD01C1  MOD01C2  MOD01C3
          LMOD        = LMOD01
          SYSMOD HISTORY = SYSMOD  TYPE      DATE    MCS    --STATUS--
                          JXY1102  FUNCTION  90.100  MOD    APP    ACC

MOD02    LASTUPD      = JXY1102 TYPE=ADD
          LIBRARIES   = DISTLIB=AOS12
          FMID        = JXY1102
          RMID        = JXY1102
          CSECT       = MOD02C1
          LMOD        = LMOD01  LMOD02
          SYSMOD HISTORY = SYSMOD  TYPE      DATE    MCS    --STATUS--
                          JXY1102  FUNCTION  90.100  MOD    APP    ACC
```

Figure 121 (Part 1 of 2). MOD Entry: Sample LIST Output When XREF Is Specified



MOD03	LASTUPD	=	JXY1121	TYPE=UPD					
	LIBRARIES	=	DISTLIB=AOS12						
	FMID	=	JXY1121						
	RMID	=	JXY1121						
	LKED ATTRIBUTES	=	RENT	REUS	AC=1				
	UMID	=	UZ00010	UZ00014					
	LMOD	=	LMOD03						
	SYSMOD HISTORY	=	SYSMOD	TYPE	DATE	MCS	--STATUS--		
			JXY1102	FUNCTION	90.100	MOD	APP	ACC	
			JXY1121	FUNCTION	90.150	MOD	APP	ACC	
			UZ00010	PTF	90.150	ZAP	APP		
			UZ00014	PTF	90.160	ZAP	APP		
MOD04	LASTUPD	=	JXY1121	TYPE=UPD					
	LIBRARIES	=	DISTLIB=AOS12						
	ASSEMBLE								
	FMID	=	JXY1121						
	RMID	=	JXY1121						
			RMIDASM						
	LMOD	=	MOD04						
	SYSMOD HISTORY	=	SYSMOD	TYPE	DATE	MCS	--STATUS--		
			JXY1102	FUNCTION	90.100	MOD	APP	ACC	
			JXY1121	FUNCTION	90.150	ASSEM	APP	ACC	

Figure 121 (Part 2 of 2). MOD Entry: Sample LIST Output When XREF Is Specified

## UNLOAD Examples

To dump the MOD entries in UCL format, you can use the UNLOAD command. To unload all the MOD entries in a particular zone, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD  MOD                 /* Unload all MOD entries. */.
```

To unload specific MOD entries, you can use these commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD  MOD(MOD01         /* Unload only these two */
          MOD02)           /* entries. */.
```

The format of the UNLOAD output for each MOD entry is the same for both of these commands. The only difference is the number of MOD entries unloaded. The following examples show UNLOAD output for MOD entries. Figure 122 on page 734 does not have cross-zone subentries. Figure 123 on page 735 does.

## MOD Entry (Distribution and Target Zone)

```
UCLIN .
REP    MOD          ( ASSEM01 )
      LASTUPD      ( JCLIN   )
      LASTUPDTYPE  ( ADD    )
      DISTLIB      ( SYSPUNCH )
      LMOD         ( LMOD99A  LMOD99B )
      .
REP    MOD          ( ASSEM02 )
      LASTUPD      ( JCLIN   )
      LASTUPDTYPE  ( ADD    )
      DISTLIB      ( SYSPUNCH )
      LMOD         ( ASSEM02 )
      .
REP    MOD          ( MOD01   )
      LASTUPD      ( JXY1102 )
      LASTUPDTYPE  ( ADD    )
      DISTLIB      ( AOS12   )
      FMID         ( JXY1102 )
      RMID         ( JXY1102 )
      CSECT        ( MOD01C1  MOD01C2  MOD01C3 )
      LMOD         ( LMOD01   )
      .
REP    MOD          ( MOD02   )
      LASTUPD      ( JXY1102 )
      LASTUPDTYPE  ( ADD    )
      DISTLIB      ( AOS12   )
      FMID         ( JXY1102 )
      RMID         ( JXY1102 )
      CSECT        ( MOD02C1 )
      LMOD         ( LMOD01   LMOD02   )
      .
REP    MOD          ( MOD03   )
      LASTUPD      ( JXY1121 )
      LASTUPDTYPE  ( UPD    )
      DISTLIB      ( AOS12   )
      FMID         ( JXY1121 )
      RMID         ( JXY1121 )
      /* LEPARM    */ RENT     REUS     AC=1
      UMID         ( UZ00010  UZ00014 )
      LMOD         ( LMOD03   )
      .
```

Figure 122 (Part 1 of 2). MOD Entry: Sample UNLOAD Output (No Cross-Zone Sub-entries)

```

REP      MOD          ( MOD04   )
          LASTUPD     ( JXY1121 )
          LASTUPDTYPE ( UPD   )
          DISTLIB     ( AOS12   )
          ASSEMBLE
          FMID        ( JXY1121 )
          RMID        ( JXY1121 )
                   RMIDASM
          LMOD        ( MOD04   )

REP      MOD          ( MOD05   )
          LASTUPD     ( JXY0001 )
          LASTUPDTYPE ( ADD   )
          DISTLIB     ( DLIB1   )
          FMID        ( JXY0001 )
          RMID        ( JXY0001 )
          /* LEPARM   */ OL      MAXBLK(6160)

ENDUCL.
    
```

Figure 122 (Part 2 of 2). MOD Entry: Sample UNLOAD Output (No Cross-Zone Subentries)

```

UCLIN .
REP      MOD          ( CICSMOD1 )
          LASTUPD     ( HCI1703 )
          LASTUPDTYPE ( ADD   )
          DISTLIB     ( ARESLIB )
          FMID        ( HCI1703 )
          RMID        ( HCI1703 )
          LMOD        ( CICSMOD1 )
          /* CROSS-ZONE*/ XZLMODP
          XZLMOD      (
                   ( LMOD01  MVS1  )
                   ( LMOD01  MVS2  )
                   )
    
```

Figure 123 (Part 1 of 2). MOD Entry: Sample UNLOAD Output (Cross-Zone Subentries)

## MOD Entry (Distribution and Target Zone)

```
REP      MOD          ( CICSMOD2 )
        LASTUPD      ( HCI1703 )
        LASTUPDTYPE  ( ADD )
        DISTLIB      ( ARESLIB )
        FMID         ( HCI1703 )
        RMID         ( HCI1703 )
        LMOD         ( CICSMOD2 )
        /* CROSS-ZONE*/ XZLMODP
        XZLMOD       (
                    ( LMOD01 MVS1 )
                    ( LMOD01 MVS2 )
                    ( LMODAA01 MVS2 )
                    )
        .
REP      MOD          ( CICSMOD3 )
        LASTUPD      ( HCI1703 )
        LASTUPDTYPE  ( ADD )
        DISTLIB      ( ARESLIB )
        FMID         ( HCI1703 )
        RMID         ( HCI1703 )
        LMOD         ( CICSMOD2 )
        .
REP      MOD          ( CICSMOD4 )
        /* CROSS-ZONE*/ XZLMODP
        XZLMOD       (
                    ( LMOD01 MVS1 )
                    ( LMOD01 MVS2 )
                    )
        .
ENDUCL.
```

Figure 123 (Part 2 of 2). MOD Entry: Sample UNLOAD Output (Cross-Zone Subentries)

By specifying the FORFMID operand, you can reduce the number of MOD entries unloaded. When **FORFMID** is specified, SMP/E unloads a MOD entry only if its FMID matches one of the FMIDs specified on the FORFMID operand. For example, to unload MOD entries whose FMIDs are either defined in FMIDSET TP or are JXY1102, you can use these commands:

```
SET      BDY(TGT1)      /* Set to target zone. */
UNLOAD  MOD            /* Unload all MOD entries */
        FORFMID(TP     /* for the TP FMIDSET */
        JXY1102)      /* and FMID JXY1102. */
```

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the MOD entry. After the UCLIN changes are done, the MOD entry must contain at least the following subentries:

- DISTLIB
- FMID
- RMID

Otherwise, there is not enough information in the entry to process the module. If any of the required subentries are missing, SMP/E does not make the requested

UCL updates to the entry, and the entry remains as it was before the UCL command.

The following examples are provided to help you use the MOD entry:

- “Example 1: Adding a New MOD Entry”
- “Example 2: Forcing Assembly of a Module” on page 738
- “Example 3: Completing Cross-Zone Updates” on page 738

### Example 1: Adding a New MOD Entry

To create a MOD entry, you should use one of the following methods rather than UCLIN:

- JCLIN, if a new module is being added to a new load module. For examples of how JCLIN creates MOD entries, see “Examples” on page 173.
- The LMOD operand on the ++MOD statement, if a new module is being added to an existing load module and no changes are needed in the link-edit control statements other than the INCLUDE for the new module. For an example of how to do this, see “++MOD MCS” on page 563.

You could make the same changes with the UCLIN command. However, you should make sure the changes are correct and complete. For example, assume you want to define two new modules, MOD99A and MOD99B. These are linked together to form load module LMOD99AB, which has an entry point of MOD99A and exists in LINKLIB. The FMID of both new modules is ZUSR001, a user-created function. You could use these commands to create the MOD and LMOD entries:

```

SET      BDY(TGT1)           /* Set to target zone.      */.
UCLIN                    /*                          */.
ADD      MOD(MOD99A)        /* Define new module entry. */
          DISTLIB(USRDLIB1) /* Define DLIB.             */
          LMOD(LMOD99AB)    /* Load module.            */
          FMID(ZUSR001)     /* Functional owner.       */
          /*                          */.
ADD      MOD(MOD99B)        /* Define new module entry. */
          DISTLIB(USRDLIB1) /* Define DLIB.             */
          LMOD(LMOD99AB)    /* Load module.            */
          FMID(ZUSR001)     /* Functional owner.       */
          /*                          */.
ADD      LMOD(LMOD99AB)     /* Now define LMOD.        */
          SYSLIB(LINKLIB)   /*                          */
          RENT REUS         /* Attributes.             */
++LMODIN                    /* Link statements.        */
  INCLUDE USRDLIB1(MOD99A,MOD99B)
  ENTRY MOD99A
++ENDLMODIN                  /*                          */
          /*                          */.
ENDUCL                      /*                          */.

```

### Example 2: Forcing Assembly of a Module

Assume you have a macro that is used by a source, and you want to make sure the source is always assembled when the macro is changed. You can do this by setting the ASSEMBLE indicator as follows:

```

SET      BDY(DLIB1)           /* Set to DLIB zone.      */
UCLIN                               /*                          */
ADD      MOD(SRC01)          /* MOD entry for source.  */
          ASSEMBLE           /* Always assemble it.    */
ENDUCL                               /*                          */

```

### Example 3: Completing Cross-Zone Updates

If SMP/E could not complete cross-zone updates for APPLY or RESTORE processing, you may need to use UCLIN to make the remaining changes. (SMP/E issues messages and reports to tell you what cross-zone work you may need to finish.) Here are some examples of using UCLIN to change XZLMOD subentries in MOD entries. Make sure to check the messages and reports to determine whether any additional changes are needed for cross-zone subentries in LMOD entries or TARGETZONE entries.

- **Adding XZLMOD subentries:** This example adds LMOD IEANUC01 to the IGCXXX MOD entry in zone IMS1. If this MOD entry previously had no XZLMOD subentries, SMP/E automatically sets XZLMODP to indicate that the MOD entry now contains XZLMOD subentries.

```

SET BDY(IMS1)                 /* set to IMS zone        */
UCLIN                         /* start UCLIN processing */
ADD MOD(IGCXXX)              /* identify MOD entry     */
      XZLMOD((IEANUC01,MVS1)) /* add load module from MVS zone */
ENDUCL                       /* end UCLIN processing   */

```

- **Replacing XZLMOD subentries:** This example replaces all the XZLMOD subentries in MOD entry ISPLINK with two XZLMOD subentries. If this MOD entry previously had no XZLMOD subentries, SMP/E automatically sets XZLMODP to indicate that the MOD entry now contains XZLMOD subentries.

```

SET BDY(MVS1)                 /* set to first MVS zone  */
UCLIN                         /* start UCLIN processing */
REP MOD(ISPLINK)              /* identify MOD entry     */
      XZLMOD((CICSDIAG,CICS1) /* replace list with two values */
              (IMSDIAG1,IMS1)) /*                          */
ENDUCL                       /* end UCLIN processing   */

```

- **Deleting XZLMOD subentries:** This example deletes an XZLMOD subentry from the ISPLINK subentry. If this causes the XZLMOD subentry list to become empty, SMP/E automatically turns XZLMODP off to indicate that the MOD entry no longer contains XZLMOD subentries.

```

SET BDY(MVS2)                 /* set to second MVS zone */
UCLIN                         /* start UCLIN processing */
DEL MOD(ISPLINK)              /* identify MOD entry     */
      XZLMOD((IMSDIAG2,IMS1)) /* delete IMSDIAG2 reference */
ENDUCL                       /* end UCLIN processing   */

```

---

## MTSMAC Entry (SMPMTS)

The MTSMAC entry is a copy of a macro that resides only in a distribution library but is needed temporarily during APPLY processing. The MTSMAC entry is in the SMPMTS data set, which serves as a target macro library for such macros.

When SMP/E applies the SYSMODs that affect these macros, it calls utility programs to store the macros on the SMPMTS. This way, the most current service level of each macro is available for use in assemblies. After SMP/E has accepted all the SYSMODs that affect these macros, it deletes the associated MTSMAC entries from the SMPMTS.

**Note:** If you specify **SAVENTS** in the **OPTIONS** entry that is in effect during **ACCEPT** processing, SMP/E will not delete MTSMAC entries from the SMPMTS after the SYSMODs that affect those macros have been successfully accepted.

## Subentries

The MTSMAC entry contains no SMP/E data and appears to the system as a member of a normal macro library.

## LIST Examples

You cannot use SMP/E to list the MTSMAC entries. However, you can use standard system utility programs (such as IEBGENER, IEBPTPCH, and IEHLIST) or products such as ISPF to display these entries or information about the data set.

## UCLIN Examples

You can use the DEL UCL statement to delete an MTSMAC entry from the SMPMTS. This can be helpful if you plan to do an APPLY followed by ACCEPT when several target libraries have been created from the same distribution library.

When a SYSMOD is accepted into a distribution zone, the entries associated with it are automatically deleted from the SMPMTS for the RELATED target zone. However, even if the SYSMOD was also applied to other target zones created from the same distribution zone, SMP/E does not clean up the SMPMTS data sets for the other target zones.

To delete the entries from these data sets, you can accept the SYSMOD and name these other target zones as the RELATED zone. However, this would update the distribution library each time; this is time-consuming and could use up space in the distribution library data set.

Instead, you can use the DEL command to delete these entries without updating the distribution library. To determine which entries to specify, check the SMPLOG data set to see which ones SMP/E deleted during ACCEPT processing.

**Note:** You can also use the CLEANUP command to delete MTSMAC entries without specifying them individually. For more information, see Chapter 4.

### Example: Deleting an MTSMAC Entry

Assume that you have two target zones, TGT1 and TGT2, generated off the same distribution zone, DLB1. During ACCEPT processing of a SYSMOD, SMP/E has deleted MTSMAC MAC01 and MAC02 from the SMPMTS data set associated with target zone TGT2. After performing the ACCEPT, you want to delete the same macro from the SMPMTS associated with target zone TGT1. Assume either that you have a cataloged procedure for TGT1 with the correct SMPMTS specified, or that you have set up the correct DDDEF entries. The following UCLIN can be used to delete the MTSMAC entry:

```
SET      BDY(TGT1)          /* Set to TGT zone.      */.  
UCLIN                               /*                          */.  
DEL      MTSMAC(MAC01)     /* Delete the macro.     */.  
DEL      MTSMAC(MAC02)     /* Delete the macro.     */.  
ENDUCL                               /*                          */.
```

**Note:** One UCL statement is required for each MTSMAC entry to be deleted.

You can make the same changes by use of system utilities. However, the SMPLOG will not reflect the processing done.



---

## OPTIONS Entry (Global Zone)

The OPTIONS entry defines processing options that are to be used for an SMP/E command or set of commands. Although OPTIONS entries exist only in the global zone, they are also used to process commands for target and distribution zones. There are two ways you can specify that OPTIONS entry that should be in effect when you are processing a zone:

- In the GLOBALZONE, TARGETZONE, and DLIBZONE entries. The OPTIONS entry specified here is the default OPTIONS entry for that zone.
- On the SET command. The name specified on the SET command overrides the default OPTIONS name.

When SMP/E processes a command, it checks these sources to determine which OPTIONS entry should be in effect for that command. If SMP/E cannot find a reference to an OPTIONS entry, it will use defaults for some of the subentries. The following section describes these defaults, as well as the other values you can specify for each subentry.

## Subentries

These are the subentries for the OPTIONS entry as they appear in the LIST output:

### *name*

is the name of the OPTIONS entry.

The name can contain from 1 to 8 alphanumeric characters.

### **AMS**

is the name of the UTILITY entry that SMP/E is to use to obtain information when calling the access method services (AMS) utility.

The UCL operand is **AMS**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.
- If no entry name is specified, SMP/E uses a default UTILITY entry. For details, see Table 30 on page 787.

### **ASM**

is the name of the UTILITY entry that SMP/E is to use to obtain information when calling the assembler utility.

The UCL operand is **ASM**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.
- If no entry name is specified, SMP/E uses a default UTILITY entry. For details, see Table 30 on page 787.

### **COMP**

is the name of the UTILITY entry that SMP/E is to use to obtain information when calling the utility program to compress data sets.

The UCL operand is **COMP**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.
- If no entry name is specified, SMP/E uses a default UTILITY entry. For details, see Table 30 on page 787.

### **COPY**

is the name of the UTILITY entry that SMP/E is to use to obtain information when calling the copy utility.

The UCL operand is **COPY**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.
- If no entry name is specified, SMP/E uses a default UTILITY entry. For details, see Table 30 on page 787.

### **DSPREFIX**

specifies the data set prefix to be used to construct the full data set name when SMPTLIB data sets are being allocated for RELFILES. For more information about names for SMPTLIB data sets, see “Receiving SYSMODs Packaged in Relative Files” on page 243.

The UCL operand is **DSPREFIX**(*prefix*).

- The prefix can contain from 1 to 26 alphanumeric characters.
- The prefix must follow standard naming conventions for data sets.
- If a DDDEF entry is in effect for the SMPTLIB data sets, the DSPREFIX value in that DDDEF entry overrides the DSPREFIX value specified in the OPTIONS entry in effect.
- If no DSPREFIX value is specified in either the OPTIONS entry or the DDDEF entry, no high-level qualifier is assigned to the SMPTLIB data sets.
- If the DSPREFIX value is the same as the RFDSNPFX value for the RELFILE data sets that are being processed, the SMPTLIB data sets cannot be allocated in either of these cases:
  - The RELFILE data sets are on DASD.
  - The RELFILE data sets are on tape and are cataloged.

### **DSSPACE**

specifies the primary and secondary space allocation (in tracks) and the number of directory blocks to be allocated for each SMPTLIB data set. After the data set is loaded, unused space is freed. For more information about SMPTLIB data sets, see “Receiving SYSMODs Packaged in Relative Files” on page 243.

The UCL operand is **DSSPACE**(*prime,secondary,directory*).

- Each value can contain from 1 to 4 numeric characters.
- If a DDDEF entry is in effect for the SMPTLIB data sets, the SPACE and DIR values in that DDDEF entry override the DSSPACE values specified in the OPTIONS entry in effect.
- These values should be specified in the appropriate OPTIONS or DDDEF entries before you receive a relative file. Otherwise, SMP/E cannot allocate any new SMPTLIB data sets.

### **EXRTYDD**

specifies the list of ddnames that are not eligible for retry processing after an x37 abend. EXRTYDD is used with RETRYDDN in order to exclude a subset of libraries from retry processing.

The UCL operand is **EXRTYDD**(*ddname...*).

- The ddnames can contain from 1 to 8 alphanumeric characters.
- If a ddname is specified in both the EXRTYDD list and the RETRYDDN list, the ddname is excluded from retry processing.
- If no ddnames are specified for RETRYDDN, no retry processing will be done for any libraries. The ddnames specified on EXRTYDD are ignored.
- If **ALL** is specified on EXRTYDD, it is treated as just another ddname; it does **not** exclude all ddnames from retry processing. To exclude all libraries from retry processing, do **one** of the following instead:
  - Specify **RETRY(NO)** on the SMP/E command being processed.
  - Do not specify a RETRYDDN list in the OPTIONS entry that is in effect for the SMP/E command being processed.
  - Do not have an OPTIONS entry in effect for the SMP/E command being processed.

**HFSCOPY**

is the name of the UTILITY entry that SMP/E is to use to obtain information when calling the hierarchical file system (HFS) copy utility.

This is the utility used to install HFS elements and must meet the same program-to-program interface as BPXCOPY.

The UCL operand is **HFSCOPY**(*name*).

- The name can contain from 1 to 8 uppercase alphanumeric characters.
- If no HFSCOPY subentry is specified in the OPTIONS entry, SMP/E uses a default UTILITY entry. For details, see Table 30 on page 787.

**IOSUP**

is the name of the UTILITY entry that SMP/E is to use to obtain information when calling the IEHIOSUP utility program to process maintenance for a VS1 system.

The UCL operand is **IOSUP**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.

**LKED**

is the name of the UTILITY entry that SMP/E is to use to obtain information when calling the link-edit utility.

The UCL operand is **LKED**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.
- If no entry name is specified, SMP/E uses a default UTILITY entry. For details, see Table 30 on page 787.

**NOPURGE**

indicates that after SMP/E accepts SYSMODs, it should not delete the associated global zone SYSMOD entries, HOLDDATA entries, SMPPTS MCS entries, or SMPTLIB data sets.

The UCL operand is **NOPURGE**.

- The default is for **NOPURGE** not to be specified. In this case, SMP/E deletes the entries after the SYSMOD has been successfully accepted. (This is

true only if the SYSMOD has been applied and **BYPASS(APPLYCHECK)** was not specified on the ACCEPT command.)

- Although this operand can be specified in any OPTIONS entry, it is effective only when the OPTIONS entry is used during ACCEPT processing.

### **NOREJECT**

specifies that the global zone SYSMOD entry and the associated MCS entry should not be deleted after the SYSMOD is restored.

The UCL operand is **NOREJECT**.

- The default is for **NOREJECT** not to be specified. In this case, SMP/E deletes the entries after the SYSMOD has been successfully restored.
- Although this operand can be specified in any OPTIONS entry, it is effective only when the OPTIONS entry is used during RESTORE processing.

### **NUCID**

specifies a 1-digit number that is to be appended to IEANUC0 to form the name of the nucleus load module saved during APPLY processing before IEANUC01 is actually modified.

The UCL operand is **NUCID(*n*)**.

- The value can be a number from zero to nine.
- If a NUCID value is specified on the APPLY command, that value overrides the NUCID value specified in the OPTIONS entry that is in effect.
- If no NUCID value is specified in either the OPTIONS entry or on the APPLY command, no backup copy of the nucleus is created during APPLY processing.

### **PAGELEN**

specifies the page length for the SMPOUT, SMPRPT, and SMPLIST data sets.

The UCL operand is **PAGELEN(*nnnn*)**.

- The value can contain from 1 to 4 numeric characters.
- If no value is specified, the default is 60.

### **PEMAX**

specifies the maximum number of subentries that can be present in any CSI entry. Most often, the largest entry is a SYSMOD entry.

The UCL operand is **PEMAX(*nnnn*)**.

- The value can contain from 1 to 4 numeric characters, with a value range of 1 to 9999.
- If no value is specified, the default is 15000. Therefore, if you want a value higher than 9999, you must use the default.

### **RETRY**

is the name of the UTILITY entry that SMP/E is to use to obtain information when calling the utility program to compress a data set after an x37 abend.

The UCL operand is **RETRY(*name*)**.

- The name can contain from 1 to 8 alphanumeric characters.

- If no entry name is specified, SMP/E uses a default UTILITY entry. For details, see Table 30 on page 787.

### RETRYDDN

specifies the list of ddnames eligible for retry processing after an x37 abend.

The UCL operand is **RETRYDDN**(*ddname...*).

- The ddnames can contain from 1 to 8 alphanumeric characters.
- **ALL** can be specified to indicate that all ddnames are eligible.
- If no ddnames are specified, no retry processing will be done, even if **RETRY** was specified on the command being processed.
- Retry processing is attempted for partitioned data sets (PDSs) and partitioned data sets extended (PDSEs) that experience an x37 abend, indicating that they have run out of space.
- For pointers on how to set up the desired retry processing, see the *SMP/E R8.1 User's Guide*.

### SAVEMTS

indicates that MTSMAC entries should not be deleted from the SMPMTS after the SYSMODs that affect those macros have been successfully accepted.

The UCL operand is **SAVEMTS**.

- If SAVEMTS is not specified, SMP/E deletes the entries after all the SYSMODs that affect the macros have been successfully accepted. (This is true only if the SYSMOD has been applied and **BYPASS(APPLYCHECK)** was not specified on the ACCEPT command.)
- Although this operand can be specified in any OPTIONS entry, it is effective only when the OPTIONS entry is used during ACCEPT processing.

### SAVESTS

indicates that STSSRC entries should not be deleted from the SMPSTS after the SYSMODs that affect the source have been successfully accepted.

The UCL operand is **SAVESTS**.

- If SAVESTS is not specified, SMP/E deletes the entries after all the SYSMODs that affect the source have been successfully accepted. (This is true only if the SYSMOD has been applied and **BYPASS(APPLYCHECK)** was not specified on the ACCEPT command.)
- Although this operand can be specified in any OPTIONS entry, it is effective only when the OPTIONS entry is used during ACCEPT processing.

### UPDATE

is the name of the UTILITY entry that SMP/E is to use to obtain information when calling the update utility.

The UCL operand is **UPDATE**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.
- If no entry name is specified, SMP/E uses a default UTILITY entry. For details, see Table 30 on page 787.

### ZAP

specifies the name of the UTILITY entry that SMP/E is to use to obtain information when calling the superzap utility.

The UCL operand is **ZAP**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.
- If no entry name is specified, SMP/E uses a default UTILITY entry with NAME(IMASPZAP), RC(4), PRINT(SYSPRINT), and PARM().

### LIST Examples

To list all the OPTIONS entries in a global zone, you can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to requested zone.  */.  
LIST     OPTIONS          /* List all OPTIONS entries. */.
```

To list specific OPTIONS entries in a global zone, you can use these commands:

```
SET      BDY(GLOBAL)      /* Set to requested zone.  */.  
LIST     OPTIONS(OPT1,    /* List only these three   */  
          TGT1,           /* entries.                */  
          DLIB1)          /*                          */.
```

The format of the LIST output for each OPTIONS entry is the same for both of these commands. The only difference is the number of OPTIONS entries listed.

Figure 124 on page 747 is an example of LIST output for OPTIONS entries.

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
GLOBAL      OPTIONS ENTRIES

NAME
FULLOPT    AMS          = MYAMS
           ASM          = MYASM
           COMP         = MYCOMP
           COPY         = MYCOPY
           HFSCOPY      = MYHCOPY
           LKED         = MYLKED
           RETRY        = MYRETRY
           UPDATE       = MYUPDAT
           ZAP          = MYZAP
           NUCID        = 8
           PAGELEN      = 100
           PEMAX        = 5000
           NOPURGE
           NOREJECT
           SAVEMTS
           SAVESTS
           DSSPACE      = PRI=300 SEC=50 DIR=25
           DSPREFIX     = SMP.RELFILE.M9001
           RETRYDDN     = ALL
           EXRTYDD      = LINKLIB MIGLIB NUCLEUS
OPT1       ASM          = IEUASM
           HFSCOPY      = BPXCOPY
           LKED         = IEWL
           RETRY        = USERRCVR
           PAGELEN      = 120
           NOREJECT
           DSSPACE      = PRI=300 SEC=50 DIR=25
           DSPREFIX     = SMP.RELFILE.M9001
           RETRYDDN     = ALL
           EXRTYDD      = LINKLIB MIGLIB NUCLEUS
    
```

Figure 124. OPTIONS Entry: Sample LIST Output

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in an OPTIONS entry. When you use UCLIN to update an OPTIONS entry, remember that if a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.

The following examples are provided to help you use the OPTIONS entry:

- “Example 1: Connecting an OPTIONS Entry to UTILITY Entries” on page 748
- “Example 2: Changing the SMPOUT Page Length” on page 748
- “Example 3: Preparing to Receive RELFILES” on page 748
- “Example 4: Identifying Libraries for Retry Processing” on page 749

**Example 1: Connecting an OPTIONS Entry to UTILITY Entries**

Assume that you have set up three UTILITY entries: IEUASM, IEWL, and MYX37. For examples of setting up these entries, see "UTILITY Entry (Global Zone)" on page 787. Now you wish to define an OPTIONS entry, OPT1, that will be used during APPLY processing. The following UCL can be used to set up that entry:

```

SET      BDY(GLOBAL)      /* Set to global zone.    */.
UCLIN                               /*                          */.
ADD      OPTIONS(OPT1)   /* New OPTIONS entry.     */.
          ASM(IEUASM)     /* Connect to assembler data. */.
          LKED(IEWL)      /* Connect to link data.   */.
          RETRY(MYX37)    /* Connect to retry.      */.
                               /*                          */.
ENDUCL                               /*                          */.
    
```

To use the OPTIONS entry, you need either to define it as the default OPTIONS entry for the desired GLOBALZONE, DLIBZONE, or TARGETZONE entry, or to specify it on the OPTIONS operand of the SET command for the zone you want to process with these values.

**Example 2: Changing the SMPOUT Page Length**

Assume that you wish to modify the OPTIONS entry created in "Example 1: Connecting an OPTIONS Entry to UTILITY Entries" to indicate that your printer now prints 120 lines per page. The following UCL will accomplish your objectives:

```

SET      BDY(GLOBAL)      /* Set to global zone.    */.
UCLIN                               /*                          */.
ADD      OPTIONS(OPT1)   /* Add because page length is
                               not there yet.          */.
          PAGELEN(120)    /* Change page length.    */.
                               /*                          */.
ENDUCL                               /*                          */.
    
```

**Example 3: Preparing to Receive RELFILES**

To receive a new function packaged in RELFILE format, you have to define the amount of space to allocate the SMPTLIB data sets and, optionally, the prefix to be used in building the data set names. Assume that the program directory indicates that 300 tracks of primary space should be used, 50 tracks of secondary space, and 25 directory blocks, and that you want the data set names to start with *SMP.RELFILE.M9001*. The following UCL can be used to add this information to the OPTIONS entry created in "Example 1: Connecting an OPTIONS Entry to UTILITY Entries":

```

SET      BDY(GLOBAL)      /* Set to global zone.    */.
UCLIN                               /*                          */.
ADD      OPTIONS(OPT1)   /* Add because data is
                               not there yet.          */.
          DSSPACE(300,50,25) /* Space allocation.     */.
          DSPREFIX(SMP.RELFILE.M9001) /* Prefix.              */.
                               /*                          */.
ENDUCL                               /*                          */.
    
```



#### Example 4: Identifying Libraries for Retry Processing

During processing of commands that update product libraries (such as ACCEPT, APPLY, RESTORE, and LINK), utilities called by SMP/E may issue x37 abends when the libraries they are updating run out of space. SMP/E can attempt to recover from such out-of-space errors if **RETRY(YES)** was specified on the command being processed **and** if a **RETRYDDN** list is available in the **OPTIONS** entry that is in effect.

Assume that you want SMP/E to attempt this retry processing for all libraries **except** LINKLIB, MIGLIB, and NUCLEUS. The following UCL can be used to specify the desired ddnames in the **OPTIONS** entry created in “Example 1: Connecting an **OPTIONS** Entry to **UTILITY** Entries” on page 748:

```
SET      BDY(GLOBAL)           /* Set to global zone. */.
UCLIN                               /*                               */.
ADD      OPTIONS(OPT1)         /* Add because data is
                               not there yet. */
                               RETRYDDN(ALL)           /* Retry all ddnames */
                               EXRTYDD(LINKLIB,MIGLIB,NUCLEUS) /* except these. */
                               /*                               */.
ENDUCL                               /*                               */.
```

### SRC Entry (Distribution and Target Zone)

The SRC entry describes source that exists in the distribution or target libraries. (SMP/E assumes that for each SRC entry in a particular zone there exists a MOD entry with the same name.) There are two ways a SRC entry can be created:

- **Installing a SYSMOD that contains the source.** SRC entries are created the first time you install a SYSMOD that contains a ++SRC statement for source that does not yet have a SRC entry.
- **Processing JCLIN.** SRC entries can be built during JCLIN processing when SMP/E scans the assembler step and determines that the assembler input is a member of a partitioned data set.

SRC entries can also be built when SMP/E scans copy steps and finds a SELECT statement that specifies TYPE=SRC.

SMP/E records the function and service level of the source in the SRC entry, as well as information about how that source affects the structure of the distribution or target libraries and modules. Once a SRC entry exists for source, it is updated as subsequent SYSMODs that affect the source are installed.

### Subentries

These are the subentries for the SRC entry as they appear in the LIST output:

#### *name*

is the name of the source represented by the SRC entry.

The name can contain from 1 to 8 alphanumeric characters and \$, #, @, or hex C0.

#### **DISTLIB**

specifies the ddname of the distribution library for the source.

The UCL operand is **DISTLIB**(*ddname*).

The ddname can contain from 1 to 8 alphanumeric characters.

#### **FMID**

identifies the functional owner of this source. The functional owner is the last function SYSMOD that replaced this module.

The UCL operand is **FMID**(*sysmod\_id*).

The SYSMOD ID must contain 7 alphanumeric characters.

#### **LASTUPD**

identifies the cause of the last change to this SRC entry.

The UCL operand is **LASTUPD**(*value*). This subentry can contain one of the following values:

#### **JCLIN**

indicates that the change was made during JCLIN command processing.

#### **UCLIN**

indicates that the change was made as a result of UCLIN processing.

*sysmod\_id*

indicates that the change was made during the installation of the indicated SYSMOD.

The SYSMOD ID must contain 7 alphanumeric characters.

**LASTUPD TYPE**

indicates how the entry was last changed.

The UCL operand is **LASTUPDTYPE**(*value*). This subentry can contain one of the following values:

- ADD**     The entry was added.
- MOV**     The entry was moved.
- UPD**     The entry was updated.

**RMID**

identifies the last SYSMOD that **replaced** this source. Any subsequent SYSMOD that modifies this module must have a defined relationship (such as PRE or SUP) with this SYSMOD.

The UCL operand is **RMID**(*sysmod\_id*).

- The SYSMOD ID must contain 7 alphanumeric characters.
- If **RMID** is not specified but **FMID** is, SMP/E sets the RMID value to the specified FMID.

**SYSLIB**

specifies the ddname of the target library for the source.

The UCL operand is **SYSLIB**(*ddname*).

- Only one SYSLIB value can be specified.
- The ddname can contain from 1 to 8 alphanumeric characters.

**UMID**

identifies all the SYSMODs that have **updated** this source since it was last replaced. Any subsequent SYSMOD that modifies this module must have a defined relationship (such as PRE or SUP) with all these SYSMODs.

The UCL operand is **UMID**(*sysmod\_id...*).

The SYSMOD ID must contain 7 alphanumeric characters.

**LIST Examples**

To list all the SRC entries in a particular zone, you could use the following commands:

```
SET      BDY(TGT1)           /* Set to requested zone. */.
LIST     SRC                 /* List all SRC entries.   */.
```

To list specific SRC entries, you could use these commands:

```
SET      BDY(TGT1)           /* Set to requested zone. */.
LIST     SRC(SRC01           /* List only these two   */.
          SRC02)             /* entries.              */.
```

The format of the LIST output for each SRC entry is the same for both of these commands. The only difference is the number of SRC entries listed. Figure 125 is an example of LIST output for SRC entries.

## SRC Entry (Distribution and Target Zone)

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
TGT1          SOURCE ENTRIES
NAME
SRC01  LASTUPD      = JXY1102 TYPE=ADD
        LIBRARIES   = DISTLIB=ASRCLIB  SYSLIB=SRCLIB
        FMID        = JXY1102
        RMID        = JXY1102
SRC02  LASTUPD      = JXY1000 TYPE=UPD
        LIBRARIES   = DISTLIB=ASRCLIB  SYSLIB=SRCLIB
        FMID        = JXY1121
        RMID        = UZ00010
        UMID        = UZ00014  UZ00015
```

Figure 125. SRC Entry: Sample LIST Output

By specifying the FORFMID operand, you can reduce the number of SRC entries listed. When **FORFMID** is specified, SMP/E lists a SRC entry only if its FMID matches one of the FMIDs specified on the FORFMID operand. For example, to list SRC entries whose FMIDs either are defined in FMIDSET TP or are JXY1102, you could use these commands:

```
SET      BDY(TGT1)          /* Set to target zone. */.
LIST     SRC                /* List all source entries */
        FORFMID(TP        /* for the TP FMIDSET */
        JXY1102)         /* and FMID JXY1102. */.
```

You can also use the LIST command to find out the name of every SYSMOD that has modified source. To include the names of these SYSMODs in the LIST output, you can use the XREF operand, as shown in these commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
LIST     SRC                /* List all source entries */
        XREF               /* and related SYSMODs. */.
```

### Notes:

1. XREF can be used either in mass mode or in select mode.
2. SMP/E obtains the data included for the XREF operand by checking for SRC and SRCUPD entries for this module in all the SYSMOD entries. Because this data is not contained in the SRC entry itself, you cannot use UCLIN to change it in the SRC entry.

Figure 126 is an example of the LIST output produced when the XREF operand is used.

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
TGT1          SOURCE ENTRIES

NAME
SRC01  LASTUPD      = JXY1102 TYPE=ADD
        LIBRARIES   = DISTLIB=ASRCLIB  SYSLIB=SRCLIB
        FMID        = JXY1102
        RMID        = JXY1102
        SYSMOD HISTORY = SYSMOD  TYPE      DATE  MCS    --STATUS--
                          JXY1102  FUNCTION 90.100 SRC     APP    ACC

SRC02  LASTUPD      = JXY1000 TYPE=UPD
        LIBRARIES   = DISTLIB=ASRCLIB  SYSLIB=SRCLIB
        FMID        = JXY1121
        RMID        = UZ00010
        UMID        = UZ00014  UZ00015
        SYSMOD HISTORY = SYSMOD  TYPE      DATE  MCS    --STATUS--
                          JXY1102  FUNCTION 90.100 SRC     APP    ACC
                          JXY1121  FUNCTION 90.150 SRC     APP    ACC
                          UZ00010  PTF      90.150 SRC     APP
                          UZ00014  PTF      90.160 SRCUPD  APP
                          UZ00015  PTF      90.161 SRCUPD  APP
    
```

Figure 126. SRC Entry: Sample LIST Output When XREF Is Specified

## UNLOAD Examples

To dump the SRC entries in UCL format, you can use the UNLOAD command. To unload all the SRC entries in a particular zone, you could use the following commands:

```

SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD   SRC                /* Unload all SRC entries. */.
    
```

To unload specific SRC entries, you could use these commands:

```

SET      BDY(TGT1)          /* Set to requested zone. */.
UNLOAD   SRC(SRC01         /* Unload only these two */.
          SRC02)           /* entries.                */.
    
```

The format of the UNLOAD output for each SRC entry is the same for both of these commands. The only difference is the number of SRC entries listed. Figure 127 is an example of UNLOAD output for SRC entries.

```
UCLIN .
REP      SRC          ( SRC01  )
          LASTUPD     ( JXY1102 )
          LASTUPDTYPE ( ADD   )
          DISTLIB     ( ASRCLIB )
          SYSLIB      ( SRCLIB  )
          FMID        ( JXY1102 )
          RMID        ( JXY1102 )

          .
REP      SRC          ( SRC02  )
          LASTUPD     ( JXY1121 )
          LASTUPDTYPE ( UPD   )
          DISTLIB     ( ASRCLIB )
          SYSLIB      ( SRCLIB  )
          FMID        ( JXY1121 )
          RMID        ( UZ00010 )
          UMID        ( UZ00014  UZ00015 )

          .
ENDUCL .
```

Figure 127. SRC Entry: Sample UNLOAD Output

By specifying the FORFMID operand, you can reduce the number of SRC entries unloaded. When **FORFMID** is specified, SMP/E unloads a SRC entry only if its FMID matches one of the FMIDs specified on the FORFMID operand. For example, to unload SRC entries whose FMIDs either are defined in FMIDSET TP or are JXY1102, you could use these commands:

```
SET      BDY(TGT1)      /* Set to target zone.    */.
UNLOAD  SRC            /* Unload all source entries*/
          FORFMID(TP   /* for the TP FMIDSET    */.
          JXY1102)    /* and FMID JXY1102.     */.
```

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the SRC entry. After the UCLIN changes are done, the SRC entry must contain at least the following subentries:

- DISTLIB
- FMID
- RMID

Otherwise, there is not enough information in the entry to process the macro. If any of these subentries are missing, SMP/E does not make the requested UCL updates to the entry, and the entry remains as it was before the UCL command.

The following examples are provided to help you use the SRC entry:

- “Example 1: Adding a New SRC Entry” on page 755
- “Example 2: Recording the Application of a Corrective Fix” on page 755

### Example 1: Adding a New SRC Entry

Assume that you have a user application installed and want to support it with SMP/E. To do this, you must record the modules in the target and distribution zones. You should identify the MOD and LMOD entries for your application by running the JCLIN function, using the appropriate link and copy steps as input. The source elements in your application will have to be entered through UCLIN. Assume that the FMID you wish to assign the product is ZUSR001 and that no service level is to be recorded. The following UCL statements should be used for each such source:

```

SET      BDY(TGT1)          /* Set to target zone.      */
UCLIN                                         /*                          */
ADD      SRC(SRC01)         /* Define new source entry. */
          DISTLIB(ASRCLIB) /* Define DLIB,             */
          SYSLIB(SRCLIB)   /* system library          */
                               (never the SMPSTS or STS). */
          FMID(ZUSR001)    /* Functional owner (in this
                               example a user function). */
ENDUCL                                         /*                          */
SET      BDY(DLB1)         /* Now do same to DLIB.    */
UCLIN                                         /*                          */
ADD      SRC(SRC01)         /* Define new source entry. */
          DISTLIB(ASRCLIB) /* Define DLIB,             */
          SYSLIB(SRCLIB)   /* system library          */
                               (never the SMPSTS or STS). */
          FMID(ZUSR001)    /* Functional owner (in this
                               example a user function). */

```

### Example 2: Recording the Application of a Corrective Fix

Assume that you have installed a corrective fix, AZ12345, to source SRC01, outside of SMP/E, and that now you want to record that fix after it has been installed. (This is not necessary if the fix was initially installed with SMP/E.) Three updates must be made:

1. Record that the fix, AZ12345, is in the system.
2. Record that the fix is on the source.
3. Record that the source has been reassembled and installed.

## SRC Entry (Distribution and Target Zone)

---

The following UCL can be used to record these changes:

```
SET      BDY(TGT1)          /* Set to target zone.    */.  
UCLIN                               /*                          */.  
ADD      SYSMOD(AZ12345)    /* Add SYSMOD entry.      */.  
          APAR              /* Corrective fix.        */.  
          APPDATE(90100)    /* Date applied.          */.  
          APPTIME(08:00:00) /* Time applied.          */.  
          FMID(FXY1102)     /* Functional owner.      */.  
          SRCUPD(SRC01)     /* Updated SRC01.        */.  
          ASSEM(SRC01)      /* Assembled it too.     */.  
                               /*                          */.  
REP      SRC(SRC01)         /* Update SRC01 to        */.  
          UMID(AZ12345)     /* add update ID of APAR.*/.  
                               /*                          */.  
REP      MOD(SRC01)         /* Update MOD entry       */.  
          RMID(AZ12345)     /* with new replacement ID.*/.  
          RMIDASM          /* Was assembled.        */.  
                               /*                          */.  
ENDUCL                               /*                          */.  
SET      BDY(DLIB1)        /* Set to DLIB zone.     */.  
UCLIN                               /*                          */.  
ADD      SYSMOD(AZ12345)    /* Add SYSMOD entry.      */.  
          APAR              /* Corrective fix.        */.  
          RECDATE(90100)    /* Date received.         */.  
          RECTIME(08:00:00) /* Time received.         */.  
          INSDATE(90100)    /* Date accepted.         */.  
          INSTIME(09:00:00) /* Time accepted.         */.  
          FMID(FXY1102)     /* Functional owner.      */.  
          SRCUPD(SRC01)     /* Updated SRC01.        */.  
          ASSEM(SRC01)      /* Assembled it too.     */.  
                               /*                          */.  
REP      SRC(SRC01)         /* Update SRC01 to        */.  
          UMID(AZ12345)     /* add update ID of APAR.*/.  
                               /*                          */.  
REP      MOD(SRC01)         /* Update MOD entry       */.  
          RMID(AZ12345)     /* with new replacement ID.*/.  
          RMIDASM          /* Was assembled.        */.  
                               /*                          */.  
ENDUCL                               /*                          */.
```



---

## STSSRC Entry (SMPSTS)

The STSSRC entry is a copy of source that resides only in a distribution library but is needed temporarily during APPLY processing. The STSSRC entry is in the SMPSTS data set, which serves as a target source library for such modules.

When SMP/E applies the SYSMODs that affect these source, it calls utility programs to store the modules on the SMPSTS. This way, the most current service level of each module is available for use in assemblies. After SMP/E has accepted all the SYSMODs that affect these source modules, it deletes the associated STSSRC entries from the SMPSTS.

**Note:** If you specify **SAVESTS** in the OPTIONS entry that is in effect during ACCEPT processing, SMP/E will not delete STSSRC entries from the SMPSTS after the SYSMODs that affect those source have been successfully accepted.

## Subentries

The STSSRC entry contains no SMP/E data and appears to the system as a member of a normal source library.

## LIST Examples

You cannot use SMP/E to list the STSSRC entries. However, you can use standard system utility programs (such as IEBGENER, IEBPTPCH, and IEHLIST) or products such as ISPF to display these entries or information about the data set.

## UCLIN Examples

You can use the DEL UCL statement to delete an STSSRC entry from the SMPSTS. This can be helpful if you plan to do an APPLY followed by ACCEPT when several target libraries have been created from the same distribution library.

When a SYSMOD is accepted into a distribution zone, the entries associated with it are automatically deleted from the SMPSTS for the related target zone. However, even if the SYSMOD was also applied to other target zones created from the same distribution zone, SMP/E does not clean up the SMPSTS data sets for the other target zones.

To delete the entries from these data sets, you could accept the SYSMOD and name these other target zones as the related zone. However, this would update the distribution library each time; this is time-consuming and could use up space in the distribution library data set.

Instead, you can use the DEL command to delete these entries without updating the distribution library. To determine which entries to specify, check the SMPLOG data set to see which ones SMP/E deleted during ACCEPT processing.

**Note:** You can also use the CLEANUP command to delete STSSRC entries without specifying them individually. For more information, see Chapter 4.

### Example: Deleting an STSSRC Entry

Assume that you have two target zones, TGT1 and TGT2, generated off the same distribution zone, DLB1. During ACCEPT processing of a SYSMOD, SMP/E has deleted STSSRC SRC01 and SRC02 from the SMPSTS data set associated with target zone TGT2. After performing the ACCEPT, you want to delete the same source from the SMPSTS associated with target zone TGT1. Assume either that you have a cataloged procedure for TGT1 with the correct SMPSTS specified, or that you have set up the correct DDDEF entries. You can use the following UCLIN to delete the STSSRC entry:

```
SET      BDY(TGT1)           /* Set to TGT1 zone.      */.  
UCLIN                                         /*                          */.  
DEL      STSSRC(SRC01)      /* Delete the source.     */.  
DEL      STSSRC(SRC02)      /* Delete the source.     */.  
ENDUCL                                         /*                          */.
```

**Note:** One UCL statement is required for each STSSRC entry to be deleted.

You can make the same changes by using system utilities; however, the SMPLOG will not reflect the processing done.

## SYSMOD Entry (Distribution and Target Zone)

The SYSMOD entry in a distribution zone or a target zone describes a SYSMOD that has been installed in the corresponding distribution library or target library. This SYSMOD entry contains the same information as the global zone SYSMOD entry, except that it has information from only one ++VER statement, the one to install the SYSMOD.

When SMP/E installs a SYSMOD, it uses SYSMOD entries in the distribution or target zone to do the following:

- Determine the functional level of the system. SMP/E checks which function SYSMODs have been installed, and then uses that information to determine which service SYSMODs may be applicable.
- Determine the service level of the system. SMP/E checks which service SYSMODs have been installed.
- Make sure the requisites are satisfied for each SYSMOD to be installed. SMP/E checks whether a SYSMOD entry exists in the distribution or target zone for each requisite.

## Subentries

These are the subentries for the SYSMOD entry as they appear in the LIST output:

*sysmod\_id*

is the SYSMOD identification.

The SYSMOD ID must contain 7 alphanumeric characters.

### **ACCEPT**

indicates that the SYSMOD has been successfully accepted.

The UCL operand is **ACCEPT**, **ACPT**, or **ACC**.

**Note:** This subentry exists only in the distribution zone. It is required in distribution zone SYSMOD entries.

### **APAR**

indicates that this SYSMOD is an APAR, which provides a corrective fix to a problem.

The UCL operand is **APAR**.

APAR, FUNCTION, PTF, and USERMOD are mutually exclusive. If none of these operands is specified, PTF is the default.

### **APPLY**

indicates that the SYSMOD has been successfully applied.

The UCL operand is **APPLY**, **APPL**, or **APP**.

**Note:** This subentry exists only in the target zone. It is required in target zone SYSMOD entries.

### **ASSEM**

lists the assemblies done during the installation of this SYSMOD.

The UCL operand is **ASSEM(name...)**.

The name can contain from 1 to 8 alphanumeric characters.

### **BYPASS**

indicates that the BYPASS operand was specified when this SYSMOD was installed.

The UCL operand is **BYPASS**.

### **CIFREQ**

lists the conditional requisites that must be installed when this function SYSMOD is installed.

**Note:** The data specified is used by SMP/E only when present in a function SYSMOD.

The UCL operand is **CIFREQ**((*causer*, *req*)...).

- *causer* is the SYSMOD that specified this function SYSMOD on the FMID operand of an ++IF statement. *req* is the SYSMOD specified on the REQ operand as the conditional requisite associated with this function SYSMOD.
- The *causer* and *req* fields must contain 7 alphanumeric characters each.
- The CIFREQ operand is mutually exclusive with all other UCL operands. It will not cause LASTUPD and LASTUPDTYPE to be updated.

### **DELBY**

specifies the SYSMOD that deleted this SYSMOD.

The UCL operand is **DELBY**(*sysmod\_id*).

- The SYSMOD ID must contain 7 alphanumeric characters.
- This subentry is valid only for function SYSMODs.
- The DELBY operand is mutually exclusive with all other UCL operands.

### **DELETE**

lists the SYSMODs deleted by this SYSMOD.

The UCL operand is **DELETE**(*sysmod\_id*...).

The SYSMOD ID must contain 7 alphanumeric characters.

### **DELLMOD**

indicates that the SYSMOD contained a ++DELETE statement.

The UCL operand is **DELLMOD**.

### **DLMOD**

lists the load modules deleted by ++DELETE statements contained in this SYSMOD.

The UCL operand is **DLMOD**(*name*...).

### *element*

lists the data element replacements contained in the SYSMOD.

The UCL operand is *element*(*name*...).

- The name can contain from 1 to 8 alphanumeric characters.
- In place of *element*, specify one of the replacement values shown in Table 26 on page 523.
- Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *element*

operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *element* operand does not contain any *xxx* value.) Table 27 on page 526 shows the *xxx* values and the languages they represent.

### **ELEMMOV**

indicates that the SYSMOD contained a ++MOVE statement.

The UCL operand is **ELEMMOV**.

### **EMOVE**

lists the elements and load modules that were moved by ++MOVE statements contained in this SYSMOD.

The UCL operand is **EMOVE**(*name...*).

### **ERROR**

indicates that an error has occurred during the processing of this SYSMOD.

The UCL operand is **ERROR**.

- This operand can also be specified as **ERR**.
- If the RESTORE subentry is set, the RESDATE operand **must** be specified, and the RESTIME and ERROR operands **should** be specified (SMP/E will automatically set it otherwise).

### **FESN**

identifies the field engineering (FE) service number.

The UCL operand is **FESN**(*string*).

The string must contain 7 alphanumeric characters.

### **FMID**

identifies the function SYSMOD to which this SYSMOD is applicable.

The UCL operand is **FMID**(*sysmod\_id*).

The SYSMOD ID must contain 7 alphanumeric characters.

### **FUNCTION**

indicates that this SYSMOD is a function, which introduces a new function into the system.

The UCL operand is **FUNCTION**.

APAR, FUNCTION, PTF, and USERMOD are mutually exclusive. If none of these operands is specified, PTF is the default.

### **HFS**

lists the HFS element replacements (++HFS statements) in the SYSMOD.

The UCL operand is **HFS**(*name...*).

The name can contain 1 to 8 uppercase alphabetic, numeric, or national (\$, #, @) characters.

### **IFREQ**

lists the conditional requisites that were installed with this SYSMOD.

The UCL operand is **IFREQ**(*sysmod\_id...*).

### INSTALLDATE

specifies the date on which this SYSMOD was installed.

The UCL operand is **INSTALLDATE**(*yyddd*).

- This operand can also be specified as **INSDATE**. For a distribution zone SYSMOD entry, it can be specified as **ACCDATE**. For a target zone SYSMOD entry, it can be specified as **APPDATE**.
- The *yyddd* must contain 5 numeric characters. SMP/E does not check whether the specified numbers are valid.
- For the distribution zone SYSMOD entry, **INSTALLDATE** is the date the SYSMOD was accepted. For the target zone SYSMOD entry, **INSTALLDATE** is the date the SYSMOD was applied.

### INSTALLTIME

specifies the time at which this SYSMOD was installed.

The UCL operand is **INSTALLTIME**(*hh:mm:ss*).

- This operand can also be specified as **INSTIME**. For a distribution zone SYSMOD entry, it can be specified as **ACCTIME**. For a target zone SYSMOD entry, it can be specified as **APPTIME**.
- The *hh*, *mm*, and *ss* must contain 2 numeric characters each. The “:” must be coded as specified. SMP/E does not check whether the specified numbers are valid.
- For a distribution zone SYSMOD entry, **INSTALLTIME** is the time the SYSMOD was accepted. For a target zone SYSMOD entry, **INSTALLTIME** is the time the SYSMOD was applied.

### JCLIN

indicates that the SYSMOD contained inline JCLIN.

The UCL operand is **JCLIN**.

### LASTSUP

specifies the last SYSMOD that superseded this SYSMOD.

The UCL operand is **LASTSUP**(*sysmod\_id*).

The SYSMOD ID must contain 7 alphanumeric characters.

### LASTUPD

identifies the cause of the last change to this entry.

The UCL operand is **LASTUPD**(**UCLIN**), indicating that the change was made as a result of UCLIN processing.

### LASTUPDTYPE

identifies the last type of update made to this entry.

The UCL operand is **LASTUPDTYPE**(*value*). This subentry can contain one of the following values:

- |            |                        |
|------------|------------------------|
| <b>ADD</b> | The entry was added.   |
| <b>UPD</b> | The entry was updated. |

**MAC**

lists the macro replacements (++MAC statements) in the SYSMOD.

The UCL operand is **MAC**(*name...*).

The name can contain from 1 to 8 alphanumeric characters.

**MACUPD**

lists the macro updates (++MACUPD statements) in the SYSMOD.

The UCL operand is **MACUPD**(*name...*).

The name can contain from 1 to 8 alphanumeric characters.

**MOD**

lists the module replacements (++MOD statements) in the SYSMOD.

The UCL operand is **MOD**(*name...*).

The name can contain from 1 to 8 alphanumeric characters.

**NPRE**

lists negative prerequisite SYSMODs—that is, SYSMODs that must not be present in the system at the same time as this SYSMOD.

The UCL operand is **NPRE**(*sysmod\_id...*).

- The SYSMOD ID must contain 7 alphanumeric characters.
- This subentry is valid only for function SYSMODs.

**PRE**

lists the prerequisite SYSMODs—that is, SYSMODs that must be present before this SYSMOD can be installed.

The UCL operand is **PRE**(*sysmod\_id...*).

The SYSMOD ID must contain 7 alphanumeric characters.

**PTF**

indicates that this SYSMOD is a PTF, which provides preventive service fixes.

The UCL operand is **PTF**.

APAR, FUNCTION, PTF, and USERMOD are mutually exclusive. If none of these operands is specified, **PTF** is the default.

**RECDATE**

specifies the date on which this SYSMOD was received.

The UCL operand is **RECDATE**(*yyddd*).

The *yyddd* must contain 5 numeric characters. SMP/E does not check whether the specified numbers are valid.

**RECTIME**

specifies the time at which this SYSMOD was received.

The UCL operand is **RECTIME**(*hh:mm:ss*).

The *hh*, *mm*, and *ss* must contain 2 numeric characters each. The “:” must be coded as specified. SMP/E does not check whether the specified numbers are valid.

### REGEN

indicates how the SYSMOD was installed in the target libraries.

- In a DLIB zone SYSMOD entry, REGEN is not important. It is automatically set for all SYSMODs when they are accepted.
- In a target zone SYSMOD entry, if REGEN is set, it indicates that SYSGEN was used to install the SYSMOD in the target libraries. (The REGEN indicator was carried over when the distribution zone was copied into the target zone. This is generally done as part of SYSGEN.)

If REGEN is not set, it indicates that the APPLY command was used to install the SYSMOD in the target libraries.

The UCL operand is **REGEN** or **RGN**.

### RENLMOD

indicates that the SYSMOD contained a ++RENAME statement.

The UCL operand is **RENLMOD**.

### REQ

lists requisite SYSMODs—that is, SYSMODs that must be installed concurrent with this SYSMOD.

The UCL operand is **REQ**(*sysmod\_id...*).

The SYSMOD ID must contain 7 alphanumeric characters.

### RESDATE

specifies the date on which this SYSMOD was restored.

The UCL operand is **RESDATE**(*yyddd*).

- This subentry exists only in the target zone.
- The *yyddd* must contain 5 numeric characters. SMP/E does not check whether the specified numbers are valid.
- If a SYSMOD is marked “RESTORE,” the RESDATE operand **must** be specified, and the RESTIME and ERROR operands **should** be specified (SMP/E will automatically set it otherwise).

### RESTIME

specifies the time that this SYSMOD was restored.

The UCL operand is **RESTIME**(*hh:mm:ss*).

- This subentry exists only in the target zone.
- The *hh*, *mm*, and *ss* must contain 2 numeric characters each. The “:” must be coded as specified. SMP/E does not check whether the specified numbers are valid.
- If a SYSMOD is marked “RESTORE,” the RESDATE operand **must** be specified, and the RESTIME and ERROR operands **should** be specified (SMP/E will automatically set it otherwise).

### RESTORE

indicates that a RESTORE attempt has been made for this SYSMOD. The RESTORE was not successful; otherwise, the SYSMOD entry would have been deleted.



The UCL operand is **RESTORE**.

- This subentry exists only in the target zone.
- This operand can also be specified as **REST** or **RES**.
- If a SYSMOD is marked “RESTORE,” the RESDATE operand **must** be specified, and the RESTIME and ERROR operands **should** be specified (SMP/E will automatically set it otherwise).

### REWORK

identifies the level of the SYSMOD, which was received again for minor changes.

The UCL operand is **REWORK**(*level*).

- Up to 8 numeric characters can be specified.
- For SYSMODs supplied by IBM, the REWORK level is *yyyyddd*, where *yyyy* is the year the SYSMOD was reworked and *ddd* is the Julian date.
- SMP/E does not check whether this data is valid.

### RLMOD

indicates the load modules renamed by ++RENAME statements in this SYSMOD.

The UCL operand is **RLMOD**(*name...*).

### SOURCEID

lists the character strings assigned to this SYSMOD during RECEIVE or CONVERT processing. These values may have been specified by the user on the RECEIVE command, included inline on the ++ASSIGN statement, or assigned by SMP/E after conversion.

The UCL operand is **SOURCEID**(*source\_id...*).

The source ID can contain from 1 to 8 alphanumeric characters.

### SRC

lists the source replacements (++SRC statements) in the SYSMOD.

The UCL operand is **SRC**(*name...*).

The name can contain from 1 to 8 alphanumeric characters.

### SRCUPD

lists the source updates (++SRCUPD statements) in the SYSMOD.

The UCL operand is **SRCUPD**(*name...*).

The name can contain from 1 to 8 alphanumeric characters.

### SUPBY

lists the SYSMODs that superseded this SYSMOD. For functions, this includes SYSMODs that both deleted and superseded this SYSMOD.

**Note:** The SUPBY field may appear as “SUPBY(IN SYSMD).” For example, this is the case if the superseding SYSMODs were installed separately instead of on the same APPLY or ACCEPT command.

The UCL operand is **SUPBY**(*sysmod\_id...*).

- This operand can also be specified as **SUP**.

- The SYSMOD ID must contain 7 alphanumeric characters.

### **SUPING**

lists the SYSMODs superseded by this SYSMOD.

The UCL operand is **SUPING**(*sysmod\_id...*).

The SYSMOD ID must contain 7 alphanumeric characters.

### **SZAP**

lists the module superzaps (++ZAP statements) in the SYSMOD.

The UCL operand is **SZAP**(*name...*).

The name can contain from 1 to 8 alphanumeric characters.

### **UCLDATE**

specifies the date on which this SYSMOD was last modified through UCLIN.

The UCL operand is **UCLDATE**(*yyddd*).

The *yyddd* must contain 5 numeric characters. SMP/E does not check whether the specified numbers are valid.

### **UCLTIME**

specifies the time that this SYSMOD was last modified through UCLIN.

The UCL operand is **UCLTIME**(*hh:mm:ss*).

The *hh*, *mm*, and *ss* must contain 2 numeric characters each. The “:” must be coded as specified. SMP/E does not check whether the specified numbers are valid.

### **USERMOD**

indicates that this SYSMOD is a USERMOD, which puts a user modification in the system.

The UCL operand is **USERMOD**.

APAR, FUNCTION, PTF, and USERMOD are mutually exclusive. If none of these operands is specified, PTF is the default.

### **VERNUM**

specifies the relative number of the ++VER statement used when this SYSMOD was installed.

The UCL operand is **VERNUM**(*nnn*).

- *nnn* can contain from 1 to 3 numeric characters.
- When updating an existing entry, you should not specify **VERNUM**. This causes SMP/E to assume the same **VERNUM** value as in the current entry.
- If you do not specify **VERNUM** when adding a new entry, SMP/E assumes a **VERNUM** value of 1.

- The VERNUM values are kept in each entry built from information from the ++VER statements. (For example, subentries such as PRE and REQ have unique VERNUM values.) If all entries do not have the same VERNUM, an error will result.

**VERSION**

lists the function SYSMODs that are versioned by this SYSMOD. Versioning indicates that, if there are any elements in common between the SYSMODs listed and this SYSMOD, this SYSMOD's elements are at a higher functional level, and are thus the ones that should be installed.

The UCL operand is **VERSION**(*sysmod\_id...*).

The SYSMOD ID must contain 7 alphanumeric characters.

**XZAP**

lists the module superzaps in the SYSMOD (++)ZAP statements) that contain an EXPAND statement (indicating that the module should be expanded before it is updated).

The UCL operand is **XZAP**(*name...*).

The name can contain from 1 to 8 alphanumeric characters.

**LIST Examples**

To list all the SYSMOD entries in a particular zone, you could use the following commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
LIST     SYSMOD             /* List all SYSMOD entries. */.
```

To list specific SYSMOD entries, you could use these commands:

```
SET      BDY(TGT1)          /* Set to requested zone. */.
LIST     SYSMOD(UZ00001     /* List only these two    */.
          UZ00002)         /* entries.                */.
```

The format of the LIST output for each SYSMOD entry is the same for both of these commands. The only difference is the number of SYSMOD entries listed. Figure 128 on page 768 and Figure 129 on page 769 are examples of LIST output for SYSMOD entries.

# SYSMOD Entry (Distribution and Target Zone)

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
DLIB1      SYSMOD ENTRIES

NAME

HZY1102  TYPE          = DELETED
        DELBY         = HZY2102

HZY2102  TYPE          = FUNCTION
        STATUS        = REC ACC
        FMID          = HZY2102
        JCLIN         = YES
        FESN          = 1234567
        DATE/TIME REC = 90.100 08:00:00
        INS           = 90.102 08:08:00
        DELETE VER(001) = HZY1102 JZY1121 JZY1122
        NPRES VER(001) = HZZ1102
        SUPING VER(001) = AZ11111 AZ11112 AZ11113
        VERS VER(001)  = HYY1102 JYY1121 JYY1122 JYY1123
        MAC            = MAC01 MAC02 MAC03
        MOD            = MOD01 MOD02 MOD03 MOD04
        SRC            = SRC01 SRC02

HZY2121  TYPE          = FUNCTION
        STATUS        = REC ACC
        FMID          = HZY2121
        JCLIN         = YES
        FESN          = 1234567
        DATE/TIME REC = 90.100 08:30:00
        INS           = 90.108 08:38:00
        DELETE VER(001) = HZY1102 JZY1121 JZY1122
        FMID VER(001)  = HZY2102
        SUPING VER(001) = AZ11121 AZ11122 AZ11123
        MAC            = MAC01
        MOD            = MOD01 MOD02
        SRC            = SRC01 SRC03 SRC04

JZY1121  TYPE          = DELETED
        DELBY         = HZY2102

JZY1122  TYPE          = DELETED
        DELBY         = HZY2102
    
```

Figure 128. SYSMOD Entry: Sample LIST Output for a Distribution Zone

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT

TGT1          SYSMOD ENTRIES

NAME

AZ99001  TYPE          = APAR
        STATUS        = REC APP
        FMID          = HZY2102
        DATE/TIME REC = 90.100 08:00:00
                INS   = 90.100 08:08:00
        LASTSUP       = UZ00010
        ZAP           = MOD01

AZ99002  TYPE          = SUPERSEDED
        LASTSUP       = UZ00010

LUS0001  TYPE          = USERMOD
        STATUS        = REC APP
        FMID          = HZY2102
        DATE/TIME REC = 90.100 08:00:00
                INS   = 90.100 08:09:00
        PRE  VER(001) = UZ00010
        MACUPD        = MAC02

UZ00008  TYPE          = PTF
        STATUS        = REC APP
        FMID          = HZY2102
        DATE/TIME REC = 90.100 08:00:00
                INS   = 90.100 08:08:00
        LASTSUP       = UZ00010
        SUPBY(IN SYSMOD) = UZ00009

UZ00009  TYPE          = SUPERSEDED
        LASTSUP       = UZ00010

UZ00010  TYPE          = PTF
        STATUS        = REC APP
        FMID          = HZY2102
        DATE/TIME REC = 90.100 08:00:00
                INS   = 90.100 08:10:00
        PRE  VER(001) = UZ00008  UZ00007
        REQ  VER(001) = UZ00040
        SUPING VER(001) = AZ99001  AZ99002  UZ00009
        MAC           = MAC01
        MACUPD        = MAC02
        MOD           = MOD01
        SRCUPD        = SRC01

UZ00011  TYPE          = PTF
        STATUS        = REC APP
        FMID          = HXY2102
        DATE/TIME REC = 86.250 08:00:00
                INS   = 86.250 08:10:00
        SOURCEID      = IP09006  PUT9304  XAU3380
        PRE  VER(001) = UZ00010
        MOD           = MOD01
    
```

Figure 129. SYSMOD Entry: Sample LIST Output for a Target Zone

By specifying various operands, you can reduce the number of SYSMOD entries listed. If you specify any of these operands on the LIST command, SMP/E automatically assumes that SYSMOD entries are to be processed, regardless of whether the SYSMOD operand was also specified. If you specify more than one of these operands on the same LIST command, only the SYSMODs that meet all the

specified conditions are processed. For more information about these operands, see Chapter 10.

You can also use the LIST command to find any other SYSMODs that specify this SYSMOD in their DEL, PRE, REQ, NPRES, SUP, or VERSION lists. To include the names of these SYSMODs in the LIST output, you can use the XREF operand, as shown in these commands:

```
SET      BDY(DLIB1)          /* Set to requested zone.   */.
LIST     SYSMOD              /* List all SYSMOD entries */.
        XREF                 /* and SYSMOD that hit them.*/.
```

**Note:** XREF can be used either in mass mode or in select mode.

Figure 130 is an example of the LIST output produced when the XREF operand is used.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMLIST OUTPUT
DLIB1      SYSMOD ENTRIES

NAME
UZ12345  TYPE          = PTF
        STATUS        = REC
        FMID          = HZY2100
        DATE/TIME REC = 90.150 08:00:00
        INS           = 90.160 08:08:00
        SUPING VER(001) = AZ11111 AZ11112 AZ11113
        MAC           = MAC01  MAC02  MAC03
        MOD           = MOD01  MOD02  MOD03  MOD04
        SRC           = SRC01  SRC02
        NPRESBY (XREF) = UZ00001  UZ00002
        PREBY (XREF)  = UZ00003  UZ00004
        REQBY (XREF)  = UZ00005  UZ00006
        VERSIONBY(XREF) = UZ00007  UZ00008
        DELBY (XREF)  = UZ00009  UZ00010
        IFREQBY (XREF) = UZ00011  UZ00012
        SUPBY (XREF)  = UZ00013  UZ00014
```

Figure 130. SYSMOD Entry: Sample LIST Output When XREF Is Specified

**Note:** Some of the xxxBY subentries listed in this example are not actually valid for the type of SYSMOD shown (that is, PTF). For example, a SYSMOD does not specify VERSION, NPRES, or DELETE for a PTF SYSMOD. Those subentries are included here only to show the format of the output.

## UNLOAD Examples

To dump the SYSMOD entries in UCL format, you can use the UNLOAD command. To unload all the SYSMOD entries in a particular zone, you could use the following commands:

```
SET      BDY(TGT1)          /* Set to requested zone.   */.
UNLOAD   SYSMOD            /* Unload all SYSMOD entries.*/.
```

To unload specific SYSMOD entries, you could use these commands:

```
SET      BDY(TGT1)          /* Set to requested zone.   */.
UNLOAD   SYSMOD(UZ00001    /* Unload only these two   */.
          UZ00002)        /* entries.                 */.
```

The format of the UNLOAD output for each SYSMOD entry is the same for both of these commands. The only difference is the number of SYSMOD entries listed. Figure 131 on page 771 is an example of UNLOAD output for one SYSMOD.

```

UCLIN .
REP      SYSMOD      ( HZY2102 )
        /* TYPE      */ FUNCTION
        /* STATUS     */ ACC
        RECDATE      ( 90100 )
        RECTIME      ( 08:08:00 )
        INSDATE      ( 90101 )
        INSTIME      ( 10:00:34 )
        VERNUM       ( 002 )
        PRE          ( UZ00001 UZ00002 UZ00003 )
        REQ          ( UZ00004 UZ00005 UZ00006 )
        SUPING       ( AZ00001 AZ00002 AZ00003 )
        MOD          ( MOD01 MOD02 )
        MAC          ( MAC01 )
        SRC          ( SRC01 SRC02 SRC03 )
        .
REP      SYSMOD      ( HZY2102 )
        CIFREQ       (
                    ( UZ00087 UZ00088 )
                    ( UZ00090 UZ00090 )
                    )
        .
ENDUCL.
    
```

Figure 131. SYSMOD Entry: Sample UNLOAD Output

By specifying various operands, you can reduce the number of SYSMOD entries unloaded. If you specify any of these operands on the UNLOAD command, SMP/E automatically assumes that SYSMOD entries are to be processed. If you specify more than one of these operands on the same UNLOAD command, only those SYSMODs that meet all the specified conditions are processed. For more information about these operands, see Chapter 23.

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the SYSMOD entry. Generally, after UCLIN changes are done, the SYSMOD entry must contain at least these subentries, unless the entire entry has been deleted:

- ACCEPT or APPLY
- APAR, FUNCTION, PTF, or USERMOD
- INSTALLDATE
- RECDATE
- FMID

Otherwise, there is not enough information in the entry to process the SYSMOD. If any of the required subentries are missing, SMP/E does not make the requested UCL updates to the entry, and the entry remains as it was before the UCL command.

**Note:** The entry for a deleted SYSMOD can contain just the DELBY subentry. The entry for a superseded SYSMOD can contain just the SUPBY subentry. The entry for a SYSMOD that is both deleted and superseded can contain just the DELBY and SUPBY subentries.

The following examples are provided to help you use the SYSMOD entry:

- “Example 1: Creating a SYSMOD Entry”
- “Example 2: Removing the ERROR Indicator”

### Example 1: Creating a SYSMOD Entry

Assume that you have installed an APAR fix outside of SMP/E and now wish to get that fix recorded in the distribution zone. The APAR number was AZ12345, and was an update to a source SRC01. The following UCL should be used:

```
SET      BDY(DLIB1)          /* Set to DLIB zone.      */.  
UCLIN                               /*                          */.  
ADD      SYSMOD(AZ12345)     /* Specify SYSMOD.        */.  
          SRCUPD(SRC01)      /* Changed this source.    */.  
          RECDATE(90100)     /* Received on this date.  */.  
          INSDATE(90100)     /* Accepted same day.      */.  
          ASSEM(SRC01)       /* Assembled source.       */.  
          FMID(FZY2102)      /* Functional owner.       */.  
                               /*                          */.  
ADD      SRC(SRC01)          /* Now update source entry */.  
          UMID(AZ12345)      /* with update ID.        */.  
                               /*                          */.  
REP      MOD(SRC01)         /* Now update MOD         */.  
          RMID(AZ12345)      /* with new replacement ID */.  
          RMIDASM            /* from assembly.         */.  
                               /*                          */.  
ENDUCL                               /*                          */.
```

### Example 2: Removing the ERROR Indicator

Assume that during an APPLY you encountered an error that caused a PTF to be marked as in error. After looking at the output you determine that the PTF actually installed correctly, and rather than reinstall the PTF you decide to make the appropriate changes to make the PTF look installed. The following UCL can be used:

```
SET      BDY(TGT1)          /* Set to target zone.     */.  
UCLIN                               /*                          */.  
DEL      SYSMOD(UZ12345)     /* Specify SYSMOD.        */.  
          RESTORE            /* Delete restore.         */.  
          RESDATE()          /* Delete restore date.    */.  
          RESTIME()         /* Delete restore time.    */.  
                               /*                          */.  
ADD      SYSMOD(UZ12345)     /* Specify SYSMOD.        */.  
          APPLY              /* Add apply info.         */.  
          INSDATE(90100)     /*                          */.  
          INSTIME(08:00:00) /*                          */.  
                               /*                          */.  
ENDUCL                               /*                          */.
```



**Note:** This method is very prone to errors. The above example gets the SYSMOD marked as having been applied; however, you have not made all the changes necessary to get the rest of the target zone entries coordinated. Those changes include updating the RMID and UMID fields of all the elements affected by the PTF, storing superseded SYSMOD entries, and updating the global zone SYSMOD entry.

If you are not extremely familiar with SMP/E internals and how to complete the update process, the recommended method is to reapply the PTF.

### SYSMOD Entry (Global Zone)

The global zone SYSMOD entry describes a SYSMOD that exists as an MCS entry in the SMPPTS. It contains information that SMP/E obtained when it received that SYSMOD. Because SMP/E processing is designed to keep the global zone SYSMOD entry and the MCS entry synchronized, you should use only SMP/E commands to update these entries. If you use system utilities to update the MCS entries, you will get unpredictable results when processing the corresponding SYSMOD. For more information about MCS entries, see “MCS Entry (SMPPTS)” on page 721.

When SMP/E processes SYSMODs, it uses the global zone SYSMOD entry to determine which SYSMODs are applicable or whether the SYSMODs you selected are applicable. In addition, after a SYSMOD has been successfully applied or accepted, SMP/E records in the SYSMOD entry the names of the zones to which the SYSMOD was applied or accepted.

**Note:** Although this information is saved in the global zone SYSMOD entry, it is used only for reporting purposes. SMP/E does not use it during APPLY, ACCEPT, or RESTORE processing to determine the status of a SYSMOD. Instead, SMP/E uses SYSMOD entries in the target and distribution zones to determine whether a SYSMOD has been applied or accepted.

A SYSMOD entry is generally kept in the global zone until the associated SYSMOD is accepted; then the entry is deleted. You may want SMP/E to save the global zone SYSMOD entries after ACCEPT processing—for example, if you plan to do a system generation. To do this, specify NOPURGE in the OPTIONS entry that is in effect during ACCEPT processing. Likewise, you may want to save the global zone SYSMOD entries after RESTORE processing. To do this, specify NOREJECT in the OPTIONS entry that is in effect during RESTORE processing.

### Subentries

These are the subentries for the global zone SYSMOD entry as they appear in the LIST output: Only a few of these subentries can be changed with UCLIN. The description of each entry indicates whether UCLIN may be used and, if so, what the correct syntax is.

*sysmod\_id*

is the SYSMOD identification.

The SYSMOD ID must contain 7 alphanumeric characters.

#### **ACCEPT ZONE**

lists the distribution zones into which the SYSMOD has been successfully accepted.

The UCL operand is **ACCID(zone...)**.

The name can contain from 1 to 7 alphanumeric characters.

#### **APAR**

indicates that this SYSMOD is an APAR, which provides a corrective fix to a problem.

There is no UCL support for this subentry in the global zone.

**APPLY ZONE**

lists the target zones to which the SYSMOD has been successfully applied.

The UCL operand is **APPID(zone...)**.

The name can contain from 1 to 7 alphanumeric characters.

**DELETE**

lists the SYSMODs deleted by this SYSMOD.

There is no UCL support for this subentry in the global zone.

*element*

lists the data element replacements in the SYSMOD.

There is no UCL support for this subentry in the global zone.

In place of *element*, you will see one of the values shown in Table 26 on page 523.

Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *element* operand does not contain any *xxx* value.)

Table 27 on page 526 shows the *xxx* values and the languages they represent.

**ERROR**

indicates that an error occurred when this SYSMOD was received.

There is no UCL support for this subentry in the global zone.

**FESN**

identifies the field engineering (FE) service number.

There is no UCL support for this subentry in the global zone.

**FMID**

identifies the function SYSMOD to which this SYSMOD is applicable.

There is no UCL support for this subentry in the global zone.

**FUNCTION**

indicates that this SYSMOD is a function, which introduces a new function into the system.

There is no UCL support for this subentry in the global zone.

**HFS**

lists the HFS element replacements (++HFS statements) in the SYSMOD.

There is no UCL support for this subentry in the global zone.

**JCLIN**

indicates that the SYSMOD contained inline JCLIN.

There is no UCL support for this subentry in the global zone.

**MAC**

lists the macro replacements (++MAC statements) in the SYSMOD.

There is no UCL support for this subentry in the global zone.

### **MACUPD**

lists the macro updates (++MACUPD statements) in the SYSMOD.  
There is no UCL support for this subentry in the global zone.

### **MOD**

lists the module replacements (++MOD statements) in the SYSMOD.  
There is no UCL support for this subentry in the global zone.

### **NPRE**

lists negative prerequisite SYSMODs—that is, SYSMODs that must not be present in the system at the same time as this SYSMOD.  
There is no UCL support for this subentry in the global zone.

### **PRE**

lists prerequisite SYSMODs—that is, SYSMODs that must be present before this SYSMOD can be installed.  
There is no UCL support for this subentry in the global zone.

### **PTF**

indicates that this SYSMOD is a PTF, which provides preventive service fixes.  
There is no UCL support for this subentry in the global zone.

### **RECDATE**

specifies the date on which this SYSMOD was received.  
There is no UCL support for this subentry in the global zone.

### **RECTIME**

specifies the time at which this SYSMOD was received.  
There is no UCL support for this subentry in the global zone.

### **REQ**

lists requisite SYSMODs—that is, SYSMODs that must be installed concurrent with this SYSMOD.  
There is no UCL support for this subentry in the global zone.

### **REWORK**

identifies the level of the SYSMOD, which was received again for minor changes.  
For SYSMODs supplied by IBM, the REWORK level is *yyyyddd*, where *yyyy* is the year the SYSMOD was reworked and *ddd* is the Julian date.  
There is no UCL support for this subentry in the global zone.

### **SOURCEID**

lists the character strings assigned to this SYSMOD during RECEIVE or CONVERT processing. These values may have been specified by the user on the RECEIVE command, included inline on the ++ASSIGN statement, or assigned by SMP/E after conversion.

The UCL operand is **SOURCEID**(*source\_id...*).

The source ID can contain from 1 to 8 alphanumeric characters.

**SRC**

lists the source replacements (++SRC statements) in the SYSMOD.

There is no UCL support for this subentry in the global zone.

**SRCUPD**

lists the source updates (++SRCUPD statements) in the SYSMOD.

There is no UCL support for this subentry in the global zone.

**SUPING**

lists the SYSMODs superseded by this SYSMOD.

There is no UCL support for this subentry in the global zone.

**SZAP**

lists the module superzaps (++ZAP statements) in the SYSMOD.

There is no UCL support for this subentry in the global zone.

**TLIBPREFIX**

is the high-level data set name qualifier of the SMPTLIB data sets used to receive this SYSMOD, which was packaged in RELFILES. This is the DSPREFIX value that was used during RECEIVE processing.

The UCL operand is **TLIBPREFIX**(*prefix*).

- The prefix can contain from 1 to 26 alphanumeric characters.
- The prefix must follow standard naming conventions for data sets.

**Note:** If the TLIBPREFIX subentry is deleted from the entry for a SYSMOD packaged in RELFILE format, the SYSMOD cannot be applied or accepted until the TLIBPREFIX subentry is re-created with the current prefix for the SMPTLIB data sets.

**USERMOD**

indicates that this SYSMOD is a USERMOD, which puts a user modification into the system.

There is no UCL support for this subentry in the global zone.

**VERSION**

lists the function SYSMODs that are versioned by this SYSMOD. Versioning indicates that, if there are any elements in common between the SYSMODs listed and this SYSMOD, this SYSMOD's elements are at a higher functional level, and are therefore the ones that should be installed.

There is no UCL support for this subentry in the global zone.

**XZAP**

lists the module superzaps (++ZAP statements) in the SYSMOD that contain an EXPAND statement (indicating that the module should be expanded before it is updated).

There is no UCL support for this subentry in the global zone.

## LIST Examples

To list all the SYSMOD entries in a global zone, you can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to requested zone. */.
LIST     SYSMOD           /* List all SYSMOD entries. */.
```

To list specific SYSMOD entries in a global zone, you can use these commands:

```
SET      BDY(GLOBAL)      /* Set to requested zone. */.
LIST     SYSMOD(UZ00001   /* List only these two */
          UZ00002)        /* entries. */.
```

The format of the LIST output for each SYSMOD entry is the same for both of these commands. The only difference is the number of SYSMOD entries listed.

Figure 132 and Figure 133 on page 779 are examples of LIST output for SYSMOD entries.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
GLOBAL      SYSMOD ENTRIES

NAME

JXY2102  TYPE          = FUNCTION
STATUS   = REC
JCLIN    = YES
FESN     = 1234567
TLIBPREFIX = SMP.RELFILE
DATE/TIME REC = 90.100 08:00:00
APPLY ZONE = TGT1      TGT2
ACCEPT ZONE = DLIB1    DLIB2
SREL VER(001) = Z038
DELETE VER(001) = JXY1102 JXY1121 JXY1122
NPRE VER(001) = HZZ1102
SUPING VER(001) = AZ11111 AZ11112 AZ11113
VERS VER(001) = HYY1102 JYY1121 JYY1122 JYY1123
MAC        = MAC01    MAC02    MAC03
MOD        = MOD01    MOD02    MOD03    MOD04
SRC        = SRC01    SRC02
```

Figure 132 (Part 1 of 2). SYSMOD Entry: Sample LIST Output for a Global Zone (Example 1)

```

JXY2121  TYPE           = FUNCTION
        STATUS        = REC
        JCLIN         = YES
        FESN          = 1234567
        TLIBPREFIX    = SMP.RELFILE
        DATE/TIME REC = 90.100 08:30:00
        APPLY ZONE    = TGT1      TGT2
        ACCEPT ZONE   = DLIB1     DLIB2
        SREL  VER(001) = Z038
        DELETE VER(001) = JXY1102  JXY1121  JXY1122
        FMID  VER(001) = JXY2102
        SUPING VER(001) = AZ11121  AZ11122  AZ11123
        MAC      = MAC01
        MOD      = MOD01      MOD02
        SRC      = SRC01      SRC03      SRC04

AZ99001  TYPE           = APAR
        STATUS        = REC
        DATE/TIME REC = 90.100 08:00:00
        APPLY ZONE    = TGT1      TGT2
        ACCEPT ZONE   = DLIB1     DLIB2
        SREL  VER(001) = Z038
        FMID  VER(001) = JXY2102
        ZAP      = MOD01
    
```

Figure 132 (Part 2 of 2). SYSMOD Entry: Sample LIST Output for a Global Zone (Example 1)

```

PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT

GLOBAL      SYSMOD ENTRIES

NAME

LUS0001  TYPE           = USERMOD
        STATUS        = REC
        DATE/TIME REC = 90.100 08:00:00
        APPLY ZONE    = TGT1      TGT2
        ACCEPT ZONE   = DLIB1     DLIB2
        SREL  VER(001) = Z038
        FMID  VER(001) = JXY2102
        PRE   VER(001) = UZ00010
        MACUPD      = MAC02
    
```

Figure 133 (Part 1 of 2). SYSMOD Entry: Sample LIST Output for a Global Zone (Example 2)

UZ00010	TYPE	=	PTF	
	STATUS	=	REC	
	DATE/TIME REC	=	90.100 08:00:00	
	APPLY ZONE	=	TGT1 TGT2	
	ACCEPT ZONE	=	DLIB1 DLIB2	
	SREL VER(001)	=	Z038	
	FMID VER(001)	=	JXY2102	
	PRE VER(001)	=	UZ00008 UZ00007	
	REQ VER(001)	=	UZ00040	
	SUPING VER(001)	=	AZ99001 AZ99002 UZ00009	
	MAC	=	MAC01	
	MACUPD	=	MAC02	
	MOD	=	MOD01	
	SRC	=	SRC01	
	SRCUPD	=	SRC01	
UZ00011	TYPE	=	PTF	
	STATUS	=	REC	
	DATE/TIME REC	=	90.150 08:00:00	
	SOURCEID	=	IP09006 PUT9304 XAU3380	
	APPLY ZONE	=	TGT1 TGT2	
	ACCEPT ZONE	=	DLIB1 DLIB2	
	SREL VER(001)	=	Z038	
	FMID VER(001)	=	JXY2102	
	PRE VER(001)	=	UZ00010	
	MOD	=	MOD01	

Figure 133 (Part 2 of 2). SYSMOD Entry: Sample LIST Output for a Global Zone (Example 2)

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in a SYSMOD entry. When you use UCLIN to update a SYSMOD entry, remember that if a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.

The following entries are provided to help you use the SYSMOD entry:

- “Example 1: Changing the SOURCEID of a SYSMOD”
- “Example 2: Indicating That a SYSMOD Was Applied” on page 781

### Example 1: Changing the SOURCEID of a SYSMOD

Assume that you received a SYSMOD and assigned it a SOURCEID value of DATALINK, and that you now wish to change the SOURCEID value to PUT9301 so that the SYSMOD will be installed with that service level. The following UCL can be used to change the SOURCEID value:

```

SET      BDY(GLOBAL)      /* Set to global zone.    */
UCLIN   /*                  /*
REP      SYSMOD(UZ00001)  /* Specify SYSMOD.      */
          SOURCEID(PUT9301) /* Change SOURCEID value.*/
          /*                  /*
ENDUCL  /*                  /*
    
```



**Example 2: Indicating That a SYSMOD Was Applied**

Assume that you have received some service, accepted it with the BYPASS(APPLYCHECK) operand, and then performed a system generation. That service is now actually in the target libraries, and you would like that recorded in the global zone SYSMOD entry. Assume that the name of your target zone is TGT1. The following UCL can be used:

```

SET      BDY(GLOBAL)      /* Set to global zone.      */.
UCLIN                               /*                          */.
ADD      SYSMOD(UZ00001)  /* Specify SYSMOD.         */.
          APPID(TGT1)     /* Applied to this zone.   */.
                               /*                          */.
ENDUCL                               /*                          */.

```

**Note:** One UCL statement is required for each SYSMOD.

### TARGETZONE Entry (Target Zone)

The TARGETZONE entry contains information SMP/E uses to process a specific target zone and the associated target libraries. It is created by UCLIN and must be defined before you can do any other processing for that target zone.

#### Subentries

These are the subentries for the TARGETZONE entry as they appear in the LIST output:

##### *name*

is the name of the target zone. You assign the name when the zone is created.

The name can contain from 1 to 7 alphanumeric characters (A–Z, 0–9) or national characters (\$, #, @).

##### **OPTIONS**

is the name of the OPTIONS entry in the global zone that should be used in processing this target zone. For more information, see “OPTIONS Entry (Global Zone)” on page 741.

The UCL operand is **OPTIONS**(*name*).

- The name can contain from 1 to 8 alphanumeric characters.
- This name can be overridden by using the OPTIONS parameter on the SET command. For more information, see Chapter 21.
- If no OPTIONS entry name is specified, SMP/E uses a set of default utility values when processing this target zone. For more information, see “OPTIONS Entry (Global Zone)” on page 741.

##### **RELATED**

is the name of the distribution zone to which this target zone is related. A target zone is related to the distribution zone used to build the target libraries, such as during system generation.

The UCL operand is **RELATED**(*zone*).

- The zone name can contain from 1 to 7 alphanumeric characters.
- Although the entry can be defined without this subentry, you **must** define the subentry before you can install any SYSMODs in the target libraries.

##### **SREL**

lists the system releases to be supported in this target zone.

The UCL operand is **SREL**(*srel...*).

- The SREL must contain 4 characters, usually 1 alphabetic character followed by 3 numeric characters. These are the SRELS defined by IBM:

<b>System</b>	<b>SREL</b>
MVS	Z038
CICS	C150
NCP	P004
IMS, DB2	P115

- Although the entry can be defined without this subentry, you **must** define the subentry before you can install any SYSMODs in the target libraries.

**Note:** Although you can support multiple products with different SREL values from one target zone, those products are still subject to all other restrictions related to combining products in one zone. The most common reason for not being able to combine products is common element names. For example, modules or macros with the same name are found in both products, but reside in different libraries.

### TIEDTO

specifies other target zones that either:

- Supplied modules for load modules controlled by this target zone
- Control load modules that have been link-edited with modules supplied by this target zone

Any zone updated with the LINK command or cross-zone information **cannot** be processed by SMP/E Release 6 or earlier.

The UCL operand is **TIEDTO**(*zone...*).

**Note:** TIEDTO subentries are added automatically during LINK command processing. However, they are **never** automatically deleted.

### XZLINK

specifies whether APPLY and RESTORE processing in another zone should automatically update load modules in this zone when cross-zone modules previously added to those load modules by the LINK command are changed.

The UCL operand is **XZLINK**(*value*).

- This subentry can contain one of the following values:

#### DEFERRED

Cross-zone load modules controlled by this zone should **not** be automatically updated when modules previously included in them by the LINK command are updated or deleted.

To make sure that the modules are synchronized with the cross-zone load modules and that the cross-zone information in SMP/E entries is correct, some combination of the following commands must be run later (depending on how the module changes affect the load modules):

- LINK command, to include modules in the affected load modules
- Link-edit (outside of SMP/E), to delete modules from the affected load modules
- UCLIN command, to update cross-zone subentries as necessary

#### AUTOMATIC

Cross-zone load modules controlled by this zone should be automatically updated when modules previously included in them by the LINK command are updated or deleted.

- If XZLINK is not specified, SMP/E uses the default value of DEFERRED.
- XZLINK does not affect processing of the LINK command.

- The XZLINK(DEFERRED) value is listed only when the TARGETZONE entry contains TIEDTO records.

### ZDESC

is a user-written description for this zone.

The UCL operand is **ZONEDESCRIPTION**(*text*).

- The zone description can be in single-byte characters (such as English alphanumeric characters) or in double-byte characters (such as Kanji).
- The zone description can contain up to 500 bytes of data, including blanks. (For double-byte data, the 500-byte maximum includes all shift-in and shift-out characters, as well as the double-byte characters.) Extra blanks are deleted. All data beyond column 72 is ignored, including blanks.
- The zone description cannot be only blanks.
- If parentheses are included in the text, they must be in matched pairs.

## LIST Examples

To list the TARGETZONE entry for a particular target zone, you can use the following commands:

```
SET      BDY(TGT1)          /* Set to requested target. */.  
LIST     TARGETZONE        /* List TARGETZONE entry.  */.
```

Figure 134 is an example of LIST output for a TARGETZONE entry.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT  
TGT1      TZONE ENTRY  
  
NAME  
TGT1      TZONE          = TGT1  
          ZDESC          = ZONE DESCRIPTION FOR TGT1 ZONE  
          RELATED        = DLIB1  
          SREL           = P115      R020      Z038  
          OPTIONS        = OPTTGT1  
          XZLINK         = AUTOMATIC  
          TIEDTO         = CICS1     IMS1
```

Figure 134. TARGETZONE Entry: Sample LIST Output

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in the TARGETZONE entry. When you use UCLIN to update a TARGETZONE entry, remember that if a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.

The following examples are provided to help you use the TARGETZONE entry:

- “Example 1: Defining a TARGETZONE Entry” on page 785
- “Example 2: Formatting a Zone Description” on page 785

**Example 1: Defining a TARGETZONE Entry**

Assume that you are about to build a new set of target libraries and thus wish to define a new target zone named TGT2. The distribution zone that you have done the system generation from is DLIB1, the OPTIONS entry that will be set up will be called OPTTGT2, and the target zone will contain MVS (Z038) and two additional programs, each with its own SREL value (P010 and R020). The target zone will exist in data set SMPE.SMPCSI.CSI. Use the following UCLIN to define the new zone.

```

SET      BDY(GLOBAL)          /* Set to global zone.      */.
UCLIN                                         /* UCLIN for GZONE entry   */.
ADD      GZONE                 /* to set up                */.
          ZONEINDEX(          /* index for new zone.     */.
              (TGT2,SMPE.SMPCSI.CSI,TARGET) /* */.
              )                /* */.
          )                      /* */.
ENDUCL                                         /* End global zone update. */.
SET      BDY(TGT2)            /* Now define new zone.    */.
UCLIN                                         /* UCLIN to define it.    */.
ADD      TARGETZONE(TGT2)     /* Identify name.         */.
          OPTIONS(OPTTGT2)    /* OPTIONS entry to use.  */.
          SREL(Z038,         /* SRELS for MVS and     */.
              P010,         /* two other              */.
              R020)         /* programs.              */.
          RELATED(DLIB1)     /* Generated from DLIB.  */.
          )                      /* */.
ENDUCL                                         /* */.

```

**Note:** Even though the OPTIONS entry has not been set up yet, you can still refer to it in the TARGETZONE entry. Prior to processing this target zone the OPTIONS entry must be created. For examples of setting up the OPTIONS entries, see "OPTIONS Entry (Global Zone)" on page 741.

**Example 2: Formatting a Zone Description**

Assume that you enter the following zone description with the first line ending in column 72 and the second line starting in column 1:

```

-----+-----1-----+-----2----- ... -----5-----+-----6-----+-----7--
SET      BDY(TGT1)          /* Set to tgt zone.      */.
UCLIN                                         /* UCLIN for TZONE entry */.
ADD      TZONE(TGT1)       /* to set up.           */.
          ZDESC(          /* THIS IS THE DESCRIPTION FOR
THE TGT1 ZONE)
          )                      /* End of zone description. */.
ENDUCL                                         /* End tgt zone update.  */.

```

Because there is no blank between the word ending in column 72 and the next word starting in column 1, SMP/E will run the two together.

## TARGETZONE Entry (Target Zone)

---

Words in a zone description, even words that end in column 72, must be separated by a blank. To format the zone description in this example correctly, you can put a blank at the beginning of the second line:

```
-----+-----1-----+-----2----- ... -----5-----+-----6-----+-----7--
SET      BDY(TGT1)          /* Set to tgt zone.      */.
UCLIN    UCLIN              /* UCLIN for TZONE entry */.
ADD      TZONE(TGT1)       /* to set up.           */.
          ZDESC(           THIS IS THE DESCRIPTION FOR
          THE TGT1 ZONE)
                               /* End of zone description. */.
ENDUCL   /* End tgt zone update. */.
```

Because there is a blank explicitly coded between the word ending in column 72 and the word starting in column 1, SMP/E will not run the words together.

## UTILITY Entry (Global Zone)

The UTILITY entry contains information that SMP/E uses when invoking a particular system utility program, such as the load module name of the program and the parameters that should be passed. This information is used only if the UTILITY entry is named in the OPTIONS entry that is in effect. For example, to have SMP/E use certain parameters when it calls a specific link-edit utility, you must do the following:

1. Define a UTILITY entry that names the program and specifies the parameters.
2. Define an OPTIONS entry that specifies that UTILITY entry as the one to use for the link-edit utility.
3. Put that OPTIONS entry into effect, either by specifying it on the SET command or by defining it as the default OPTIONS entry for the zone to be processed.

If the OPTIONS entry does not point to a UTILITY entry for a particular system utility, SMP/E uses default values for that utility. Table 30 lists the default values for the various types of utility programs.

Utility	NAME	RC	PRINT	PARM
Access method services	IDCAMS	0	SYSPRINT	
Assembler	IEV90	4	SYSPRINT	XREF, NOLOAD, DECK
Compress	IEBCOPY	0	SYSPRINT	
Copy	IEBCOPY	0	SYSPRINT	
Hierarchical file system copy	BPXCOPY	0	SYSPRINT (see note 3)	
Link-edit utility	IEWBLINK or IEWL (see note 1)	8	SYSPRINT	NCAL or CALL, LIST, LET, XREF (see note 2)
Maintenance on VS1	IEHIOSUP	0	SYSPRINT	
Retry after x37 abends	IEBCOPY	0	SYSPRINT	
Update	IEBUPDTE	0	SYSPRINT	Determined by SMP/E during processing
Superzap	IMASPZAP	4	SYSPRINT	
<b>Notes:</b>				
1. When the DFP level supports the binder, IEWBLINK is the default link-edit utility. When the DFP level does not support the binder, IEWL is the default.				
2. SMP/E automatically determines whether to use the NCAL or CALL parameter. NCAL is the default when the load module being link-edited does not contain a CALLLIBS subentry list; CALL is the default when it does.				
3. If SYSTSPRT is specified as the PRINT value, it is ignored and the default of SYSPRINT is used instead.				

If a UTILITY entry defines some, but not all, of these subentries, SMP/E uses the default values for the subentries not specified. For more information about OPTIONS entries, see “OPTIONS Entry (Global Zone)” on page 741.

### Notes:

1. Besides defining the information that is passed to utility programs, you can define which utility programs SMP/E is authorized to load. This is done through module GIMUTTBL. For more information, see Appendix F.
2. Because SMP/E runs authorized, all the utility programs called by SMP/E must reside in an authorized library. The utility program must be in either LINKLIB or the link pack area (LPA).

## Subentries

These are the subentries for the UTILITY entry as they appear in the LIST output:

### *name*

is the name of the UTILITY entry.

The name can contain from 1 to 8 alphanumeric characters.

### LIST

indicates whether member names should be listed when SMP/E invokes a copy utility to perform compress processing, retry processing, or element installation.

The UCL operand is **LIST(YES|NO)**.

- YES indicates that member names should be listed during copy processing. This is the default.
- NO indicates that the list of member names should be suppressed during copy processing.
- The LIST value (including the default value) is ignored when the UTILITY entry is not for a copy utility.

### NAME

is the name of the load module for the utility program that SMP/E is to call. Table 30 on page 787 lists the default names used by SMP/E if the OPTIONS entry in effect does not point to a UTILITY entry for a specific system utility.

The UCL operand is **NAME(prog)**.

The name can contain from 1 to 8 alphanumeric characters. The first character must be alphabetic.

### PARM

specifies the parameters to be passed to the utility program. SMP/E may add other parameters to this list. Table 30 on page 787 lists the default parameters used by SMP/E if the OPTIONS entry in effect does not point to a UTILITY entry for a specific system utility.

The UCL operand is **PARM(string)**.

- If you specify **PARM**, you must specify all the parameters to be passed. You cannot specify a single value to add or change just one parameter.



- The parameter string is usually a hexadecimal character string with a length of up to 100 characters. This 100-character limit can be exceeded for a binder link-edit step using the OPTION option. For a more detailed explanation, see the note under “EXEC Statement” on page 160. No validity checking is done on the string. If any blanks are specified between the parentheses, they will be deleted by SMP/E during processing.

- If the string contains parentheses, they must be in matched pairs. SMP/E assumes it has reached the end of the specified string when it encounters a closing parenthesis for the opening parenthesis. For example, the following is valid because there are matching parentheses within the string:

```
PARM(LIST,LET,SIZE=(1526K,80K),NCAL)
```

Parentheses within quotation marks are still considered parentheses. The next example is not valid because there are unmatched parentheses. SMP/E would continue scanning after the last parenthesis looking for another “)”.

```
PARM(PRINT(A),SPECIALCHAR('))
```

Likewise, the following example is not valid. SMP/E would stop at the “)” after SPECIALCHAR and give a syntax error on the closing quotation mark.

```
PARM(PRINT(A),SPECIALCHAR')')
```

- **Assembler utility:** If no PARM subentry is specified for an assembler utility, the parameters XREF, NOLOAD, and DECK are used.

If you change the PARM subentry, either include DECK in the parameters chosen or ensure that the assembler program produces an object deck.

- **HFS copy utility:** If the UTILITY entry for the HFS copy utility specifies a PARM value, those parameters are passed to the utility in addition to any parameters saved in the HFS entry.

- **Link-edit utility:** If no PARM subentry is specified for a link-edit utility, the parameters LET, LIST, XREF, and NCAL or CALL are used. SMP/E automatically selects NCAL or CALL depending on whether or not the load module being link-edited contains a CALLLIBS subentry list. NCAL is used when a load module does not contain a CALLLIBS list, and CALL is used when it does. CALL is never used for ACCEPT processing. These parameters are passed in addition to the link-edit attributes determined during APPLY and ACCEPT processing.

LET and either NCAL or CALL (as appropriate) are ordinarily required for maintenance of IBM operating systems. If you change the PARM subentry, be sure to include LET in the parameters chosen. (SMP/E automatically determines whether to use NCAL or CALL, and ignores whether either of these values was specified in the UTILITY entry.) For more information, see “Link-Edit Parameters and Load Module Attributes” on page 97.

- **Update utility:** The PARM subentry for an update utility must not specify MOD or NEW.

#### PRINT

specifies the ddname that is to contain output from the utility (for example, SYSPRINT). Table 30 on page 787 lists the default ddnames used by SMP/E if the OPTIONS entry in effect does not point to a UTILITY entry for a specific system utility.

The UCL operand is **PRINT**(*ddname*).

The value specified can contain from 1 to 8 alphanumeric characters. The first character must be alphabetic.

### Notes:

1. The *ddname* specified for PRINT can affect whether you receive print output from this utility. For example, if you specify a DDDEF for a DUMMY data set or a DDDEF for a data set that is sent to a SYSOUT class that suppresses output, you do not receive print output from this utility.
2. If SYSTSPRT is specified as the PRINT value for the HFS copy utility, it is ignored and the default of SYSPRINT is used instead.

### RC

specifies the maximum acceptable return code from this utility. If the return code is higher than this value, SMP/E normally assumes that the requested processing failed. However, the success of a link-edit in the SMPLTS library is based on the following:

- Link-edits into the SMPLTS with a return code of 8 or less are considered successful regardless of the threshold return code specified in the UTILITY entry.
- Link-edits into the SMPLTS with a return code of greater than 8 are considered successful or failures based on the threshold (normal processing). If the SMPLTS link-edit return code is less than or equal to the threshold return code, it is considered successful. If the SMPLTS link-edit return code is greater than the threshold return code, it is considered a failure.

Table 30 on page 787 lists the default return codes used by SMP/E if the OPTIONS entry in effect does not point to a UTILITY entry for a specific system utility.

The UCL operand is **RC**(*rc*).

- The value can be from 0 to 16.
- The installation information for a product (such as the product program directory) may state the utility return codes you should expect during SMP/E processing. For example, it may state the expected link-edit return codes for its load modules during SMP/E processing. The expected return code may be 4 or 8, because post-SMP/E link-edit work is required (for example, the load modules may require interface routines or compiler library routines). Such return codes allow the SYSMODs to be installed, but they also require you to check the actual link-edit return code in the GIM23903 messages in order to determine the actual success of utility processing.

Before using the default SMP/E return codes (especially for link-edit processing), check the installation information for the products you plan to install and determine the appropriate maximum acceptable return codes for utility processing. You may find that you need to set up more than one UTILITY entry for a particular utility program in order to accommodate different maximum return codes for various products. (As a result, you may need additional OPTIONS entries to point to the appropriate UTILITY entries.)

## LIST Examples

To list all the UTILITY entries in a global zone, you can

```
SET    BDY(GLOBAL)      /* Set to requested zone.  */.
LIST   UTILITY          /* List all UTILITY entries. */.
```

To list specific UTILITY entries in a global zone, you can use these commands:

```
SET    BDY(GLOBAL)      /* Set to requested zone.  */.
LIST   UTILITY(IEUASM   /* List only these three   */
        IEWL           /* entries.                */
        MYX37)         /*                          */.
```

The format of the LIST output for each UTILITY entry is the same for both of these commands. The only difference is the number of UTILITY entries listed.

Figure 135 shows an example of LIST output for UTILITY entries.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMLIST OUTPUT
GLOBAL      UTILITY ENTRIES

NAME
IEUASM  PRINT      = MYSYSPRT
IEWL    PARM       = SIZE=(1526K,80K)
MYX37   NAME       = USERRCVR
        PARM       = TYPE=FAST
        PRINT      = X37PRINT
        RC         = 4
        LIST       = NO
```

Figure 135. UTILITY Entry: Sample LIST Output

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in a UTILITY entry. When you use UCLIN to update a UTILITY entry, keep these points in mind:

- After the UCLIN changes are done, the UTILITY entry must contain at least one of the following subentries:
  - NAME
  - PARM
  - PRINT
  - RC

Otherwise, there is not enough information in the entry for SMP/E to use the entry.

- If a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.

The following examples are provided to help you use the UTILITY entry:

- “Example 1: Changing the DD Statement for SYSPRINT Output” on page 792
- “Example 2: Defining Link-Edit Utility Parameters” on page 792

- “Example 3: Defining a User Utility Program” on page 793

### Example 1: Changing the DD Statement for SYSPRINT Output

Assume that you wish to direct all the SYSPRINT output for assemblies to the MYSYSPRT DD statement rather than to the SYSPRINT DD statement. To do this you need to build a UTILITY entry for the assembler and then specify that UTILITY entry in the OPTIONS entry that will be in effect. Assume that the UTILITY entry to be defined is named IEUASM, and that the OPTIONS entry to be used is MYOPT1. The following UCL will accomplish your objectives:

```

SET      BDY(GLOBAL)          /* Set to global zone.      */
UCLIN                                /*                          */
ADD      UTILITY(IEUASM)      /* Assembler utility.       */
          PRINT(MYSYSPRT)    /* Alternate SYSPRINT      */
          /* Other values remain as
          in SMP/E defaults. */
REP      OPTIONS(MYOPT1)     /* Connect to OPTIONS.     */
          ASM(IEUASM)        /* Use IEUASM for ASM.     */
          /*                          */
ENDUCL                                /*                          */

```

**Note:** Remember, if you specify a DDDEF for a DUMMY data set or a DDDEF for a data set that is sent to a SYSOUT class that suppresses output, you do not receive SYSPRINT output from this utility.

### Example 2: Defining Link-Edit Utility Parameters

Assume that you wish to always pass the **SIZE=(1526K,80K)** parameter to the link-edit utility. To do this you need to build a UTILITY entry for the link-edit utility and then specify that UTILITY entry in the OPTIONS entry that will be in effect. Assume that the UTILITY entry to be defined is named IEWL, and that the OPTIONS entry to be used is MYOPT1. The following UCL will accomplish your objectives:

```

SET      BDY(GLOBAL)          /* Set to global zone.      */
UCLIN                                /*                          */
ADD      UTILITY(IEWL)        /* Link-edit utility.       */
          PARM(LET,           /* PARM values.            */
              SIZE=(1526K,80K), /*                          */
              NCAL)          /*                          */
          /* Other values remain as
          in SMP/E defaults. */
REP      OPTIONS(MYOPT1)     /* Connect to OPTIONS.     */
          LKED(IEWL)         /* Use IEWL for LKED.     */
          /*                          */
ENDUCL                                /*                          */

```

### Example 3: Defining a User Utility Program

Assume that you wish SMP/E to call a user routine, USERRCVR, rather than IEBCOPY, in order to recover from x37 abends. In addition to the program name change, the program must also receive parameter TYPE=FAST. You want to indicate that the output should go to X37PRINT rather than SYSPRINT, and that a return code of 4 or less is acceptable. You also want to suppress the listing of member names during retry processing done by your program. To do this you need to build a UTILITY entry for the program, and then specify the UTILITY entry in the OPTIONS entry that will be in effect. Assume that the UTILITY entry to be defined is named MYX37, and that the OPTIONS entry to be used is MYOPT1. The following UCL will accomplish your objectives:

```
SET      BDY(GLOBAL)      /* Set to global zone.    */.  
UCLIN                                       /*                          */.  
ADD      UTILITY(MYX37)   /* Retry/recovery program.*/  
          NAME(USERRCVR)  /* Program name.          */  
          PARM(TYPE=FAST) /* PARM value.            */  
          PRINT(X37PRINT) /* SYSPRINT ddname.      */  
          RC(4)           /* Highest acceptable     */  
                          /* return code.           */  
          LIST(NO)       /* No list of member names.*/  
                          /*                          */.  
REP      OPTIONS(MYOPT1)  /* Connect to OPTIONS.    */  
          RETRY(MYX37)    /* Use MYX37 for RETRY.   */  
                          /*                          */.  
ENDUCL                                       /*                          */.
```

### ZONESET Entry (Global Zone)

The ZONESET entry defines a group of zones to be used to limit the SYSMODs processed by an SMP/E command. For example, you can specify a ZONESET on the ZONESET operand of the REPORT command. This defines which zones SMP/E should check for installed SYSMODs that specify conditional requisites that might be needed for functions in other zones in the ZONESET. A ZONESET may also define a group of zones to be checked or ignored by the REJECT command.

### Subentries

These are the subentries for the ZONESET entry as they appear in the LIST output:

#### *name*

is the name of the ZONESET. The name can contain from 1 to 8 alphanumeric characters.

To avoid confusion and undesired results, you may want to avoid giving a ZONESET the same name as any of the target or distribution zones defined to the global zone that will contain the ZONESET entry. This is because, on some SMP/E command operands, you can specify zones and ZONESETs. When ZONESETs and zones have the same name, you might not get the results you wanted.

For example, suppose you have a ZONESET named MVS1 and a zone named MVS1. If you specify MVS1 on an SMP/E command operand, SMP/E assumes that you want to use the zones defined in ZONESET MVS1 (which might or might not include zone MVS1), and not the individual zone MVS1.

#### **ZONE**

lists the target or distribution zones that are to be part of this ZONESET.

The UCL operand is **ZONE(zone...)**.

- Each value can contain from 1 to 7 alphanumeric characters.
- A ZONESET can contain both target and distribution zones.
- All the zones in a ZONESET must be defined in the same global zone as the ZONESET entry.
- The zones cannot be defined in global zones that are in different SMPCSI data sets. For an example of defining a ZONESET in order to report on zones controlled by different global zones, see "Example 2: Using REPORT CROSSZONE with Zones Controlled by Different Global Zones" on page 294.

### LIST Examples

To list all the ZONESET entries in a global zone, you can use the following commands:

```
SET      BDY(GLOBAL)      /* Set to global zone.    */.  
LIST     ZONESET          /* List all ZONESET entries. */.
```

To list specific ZONESET entries in a global zone, you can use these commands:

```
SET      BDY(GLOBAL)      /* Set to global zone.    */.
LIST     ZONESET(S370     /* List only these       */.
        SESA)            /* entries.               */.
```

The format of the LIST output for each ZONESET entry is the same for both of these commands. The only difference is the number of ZONESET entries listed.

Figure 136 shows an example of LIST output for ZONESET entries.

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnn DATE mm/dd/yy TIME hh:mm:ss GIMSMP LVL 18.1.nn SMPLIST OUTPUT
GLOBAL                ZONESET ENTRIES

SESA      ZONE      = MVSESA  PRODESA  PROD370
S370     ZONE      = MVS370  PRODESA  PROD370
```

Figure 136. ZONESET Entry: Sample LIST Output

## UCLIN Examples

You can use the ADD, REP, and DEL UCL statements to change subentries in a ZONESET entry. When you use UCLIN to update a ZONESET entry, keep these points in mind:

- After the UCLIN changes are done, the ZONESET entry must contain at least a ZONE subentry. Otherwise, the entry contains so little information that SMP/E cannot use it.
- If a DEL statement deletes all the existing subentries in the entry, SMP/E deletes the entire entry.

### Example: Defining a ZONESET Entry

Assume that you have a system with four target zones: MVS370, PROD370, MVSESA, and PRODESA. MVS370 and MVSESA define two different MVS base control programs. PROD370 and PRODESA are two versions of the same product that must be synchronized with each other and with their base control programs (PROD370 with MVS370 and PRODESA with MVSESA). To keep service for these products at the same level, you can group MVS370, PROD370, and PRODESA in one ZONESET (S370) and MVSESA, PRODESA, and PROD370 in a second ZONESET (SESA). The following UCL will define the ZONESET entries:

```
SET      BDY(GLOBAL)      /* Set to global zone.    */.
UCLIN    /*                */.
ADD      ZONESET(S370)     /* ZONESET S370.         */.
        ZONE(MVS370,      /* Include these target  */.
        PROD370,         /* zones.                */.
        PRODESA)        /*                        */.
        /*                */.
ADD      ZONESET(SESA)    /* ZONESET SESA.        */.
        ZONE(MVSESA,     /* Include these target  */.
        PRODESA,         /* zones.                */.
        PROD370)        /*                        */.
        /*                */.
ENDUCL   /*                */.
```





## Chapter 36. SMP/E Installation-Wide Exit Routines

### Note

This chapter describes Product-Sensitive Programming Interface and Associated Guidance Information. For a definition of Product-Sensitive programming interfaces, see "Programming Interface Information" on page xxi.

This chapter describes how to write exit routines that:

- Process statements in SMPPTFIN at RECEIVE time
- Control retry processing when data sets run out of space at ACCEPT, APPLY, LINK, or RESTORE time

These exit routines are called by a driver (GIMMPUXD), which receives control whenever SMP/E detects that an installation-wide exit routine can be called.

1. When SMP/E reaches a point where an installation-wide exit routine can be called, it calls GIMMPUXD to see whether the appropriate routine exists.
2. GIMMPUXD checks for the indicated exit routine and passes SMP/E its name or address, or an indication that there is no routine.
3. If there is an exit routine, SMP/E passes it the appropriate information to process.
4. The exit routine processes the information and returns the appropriate output to SMP/E.

A common parameter list is used to pass information between SMP/E, the driver, and the exit routines. It is pointed to by register 1 and is mapped by macro GIMMPUXP in SYS1.MACLIB. Table 31 shows the format and contents of the parameter list.

Field Name	Offset (DEC)	Offset (HEX)	Length	Description
UXPUXNUM	+ 0	+ 0	2	Exit routine number: X'0001' – RECEIVE X'0002' – RETRY
	+ 2	+ 2	2	Not used
UXPUXNAM	+ 4	+ 4	8	Name of exit routine
UXPUXAD	+12	+ C	4	Address of exit routine
UXPFUNCT	+16	+10	8	SMP/E command
UXPPRMAD	+24	+18	4	Address of exit routine parameter list
UXPLOJAD	+28	+1C	4	Not used
UXPLOEAD	+32	+20	4	Not used
UXPCTBAD	+36	+24	4	Reserved for future use
UXPMODAD	+40	+28	4	Reserved for future use

The following sections describe the driver (GIMMPUXD) and the installation-wide exit routines for RECEIVE and retry processing.

---

### Driver for Installation-Wide Exit Routines

GIMMPUXD is called by SMP/E to check whether a requested installation-wide exit routine exists. GIMMPUXD is a separate load module that is loaded during SMP/E initialization (using the LOAD macro) and linked to during processing (using the CALL macro). A dummy version of GIMMPUXD comes with SMP/E and resides in SYS1.LINKLIB. It always passes a return code of 0 and does not change the parameter list that is passed to it. If you want to code any installation-wide exit routines, you must replace this dummy version with your own version of the module. Your version must follow these rules:

- It must reside in either SYS1.LINKLIB or another authorized library.
- It must follow standard linkage conventions:
  - The register values at invocation must be the same when the module returns to SMP/E, except for registers 0, 1, and 15.
  - The registers must be saved in an area with backward and forward save area chains.
  - The address of the parameter list must be passed in register 1. This parameter list is described in Table 31 on page 797.

When GIMMPUXD is called by SMP/E, the exit routine number is in field UXPUXNUM of the parameter list. GIMMPUXD must determine whether that exit routine exists. If so, it can activate the exit routine either by setting UXPUXNAM to the name of the exit routine (and UXPUXAD to zeros) or by setting field UXPUXAD to the entry point address of the exit routine (and UXPUXNAM to blanks). If the requested exit routine does not exist, GIMMPUXD must set UXPUXNAM to blanks and UXPUXAD to zeros.

If the driver passes the name of the exit routine, SMP/E loads the exit routine using the LOAD macro, and then invokes it using the CALL macro. When the exit routine is no longer needed, SMP/E deactivates it using the DELETE macro.

If the driver passes the address of the exit routine, SMP/E invokes the exit routine using a CALL macro. However, it does not load the exit routine. You must either issue the LOAD macro for the routine, or include it in GIMMPUXD so that it is loaded as part of that module.

In addition to the parameter list, GIMMPUXD must set register 15 to one of the following values:

Value	Meaning
0	Continue normal processing. (See parameter list for more information.)
12	Stop command processing.
16	Stop SMP/E processing.

If any other value is returned, SMP/E converts it to 12, and command processing fails.

## RECEIVE Exit Routine

The RECEIVE exit routine allows you to scan statements in the SMPPTFIN data set at RECEIVE time. It is similar to module GIMMPEXT, which was in SMP Release 4. This exit routine must be a load module residing in an authorized library. It can be in SYS1.LINKLIB, in a data set defined by the STEPLIB or JOBLIB DD statements, or part of GIMMPUXD, the driver for installation-wide exit routines. The RECEIVE exit routine is loaded at the start of RECEIVE processing and is deleted at the end of RECEIVE processing.

When this exit routine is called, the parameter list contains the values shown in Table 32 and Table 33.

*Table 32. RECEIVE Exit Routine: Parameter List Values*

Field Name	Description
UXPUXNUM	X'0001' (exit routine number)
UXPUXNAM	Name of exit routine
UXPUXAD	Address of exit routine
UXPFUNCT	RECEIVE
UXPPRMAD	Address of 81-byte buffer area (see Table 33)

*Table 33. RECEIVE Exit Routine: Buffer Passed by UXPPRMAD*

Field Name	Offset (DEC)	Offset (HEX)	Length	Description
UX001RC	+ 0	+ 0	1	X'00' – Buffer contains record to be processed X'04' – End-of-file on SMPPTFIN
UX001RCD	+ 1	+ 1	80	Record from SMPPTFIN

According to the input record, the RECEIVE exit may decide to continue RECEIVE processing, change the record, insert data after the record, or skip the record. Or, it may choose to stop processing for the SYSMOD, for the RECEIVE command, or for SMP/E.

When the exit routine returns control to SMP/E, it must set register 15 to one of the following values:

Value	Meaning
0	Continue normal RECEIVE processing.
8	Stop SYSMOD processing. SMP/E does not receive this SYSMOD, but continues to pass records from the SYSMOD to the exit routine.
12	Stop RECEIVE processing.
16	Stop SMP/E processing.
20	Insert a record after the current one in the buffer.
24	Skip the record in the buffer area.

If any other value is returned, SMP/E issues an error message and fails.

### Changing Records Within SMPPTFIN

To change the current record within the SMPPTFIN data set, the exit routine must change the record currently in UX001RCD and set the return code in register 15 to 0 or 20. A return code of 0 indicates to SMP/E that the changed record should be processed and normal processing should continue. A return code of 20 indicates to SMP/E that the exit routine inserts records after this changed record. For more information, see "Inserting Records Within SMPPTFIN."

### Inserting Records Within SMPPTFIN

To insert one or more records within the SMPPTFIN data set after the current record in UX001RCD, the exit routine must set the return code in register 15 to 20.

SMP/E first processes the current record in UX001RCD; then it calls the exit routine again. On this call, the exit routine must place the record to be inserted into UX001RCD. If more records are to be inserted, the exit routine must set the return code in register 15 to 20. If this is the last record to be inserted, the exit routine must set the return code to 0.

When the exit routine passes a return code of 0, SMP/E processes the record in UX001RCD; then it continues processing with the next record in the SMPPTFIN data set.

### Inserting Records at the End of SMPPTFIN

To insert one or more records at the end of the SMPPTFIN data set, the exit routine must set the return code in register 15 to 20.

SMP/E then calls the exit routine again. On this call, the exit routine must place the record to be inserted into UX001RCD. If more records are to be inserted, the exit routine must set the return code in register 15 to 20. If this is the last record to be inserted, the exit routine must set the return code to 0. SMP/E processes the inserted records as if they were at the end of the SMPPTFIN data set.

When the exit routine passes a return code of 0, SMP/E processes the record in UX001RCD; then it continues with normal end-of-file processing.

**Note:** When you specify the SELECT operand during RECEIVE processing, and all selected SYSMODs have been processed, SMP/E indicates end-of-file to the exit routine. If the exit routine passes a return code of 20 at this point, the add request is ignored, and SMP/E continues with end-of-file processing.

### Skipping Records in SMPPTFIN

To skip a record in SMPPTFIN, the exit routine must set the return code in register 15 to 24. SMP/E does not process the current record in UX001RCD; instead, it processes the next record in SMPPTFIN.

## Retry Exit Routine

The retry exit routine enables you to control retry processing when a data set runs out of space at ACCEPT, APPLY, LINK, or RESTORE time. (In retry processing, the data set is compressed and the utility that failed is called again.) This exit routine must be a load module that resides in an authorized library. It can be in SYS1.LINKLIB, in a data set defined by the STEPLIB or JOBLIB DD statements, or part of GIMMPUXD, the driver for installation-wide exit routines. The retry exit routine is loaded at the start of SMP/E processing and deleted at the end of SMP/E processing.

**Note:** The processing of this routine is not affected by the debatching SMP/E does after retry processing fails, because this routine is called as part of the initial retry processing, **before** debatching is attempted.

When a data set runs out of space, an x37 abend occurs. If SMP/E determines that a retry can be attempted, it cancels the abend dump and calls the retry exit routine. The parameter list contains these values:

Field Name	Description
UXPUXNUM	X'0002' (exit routine number)
UXPUXNAM	Name of exit routine
UXPUXAD	Address of exit routine
UXPFUNCT	APPLY, ACCEPT, LINK, or RESTORE
UXPPRMAD	Address of 25-byte parameter list (see Table 35)

Field Name	Offset (DEC)	Offset (HEX)	Length	Description
UX002DDN	+ 0	+ 0	8	ddname of data set that ran out of space
UX002PGM	+ 8	+ 8	8	Name of the utility program that failed
UX002ACH	+16	+10	3	ABEND code (hex) in the same format as field SDWACMPC in the SDWA control block

Table 35 (Page 2 of 2). Retry Exit Routine: Parameter List Passed by UXPPRMAD

Field Name	Offset (DEC)	Offset (HEX)	Length	Description
UX002RCH	+19	+13	1	ABEND reason code (hex)
UX002ACP	+20	+14	3	ABEND code (EBCDIC)
UX002RCP	+23	+17	2	ABEND reason code (EBCDIC)

According to the input, the retry exit can either cancel retry processing or perform some other method of recovery.

When the exit routine returns control to SMP/E, it must set register 15 to one of the following values:

Value	Meaning
0	Continue normal retry processing.
12	Stop command processing and perform no retry.
16	Stop SMP/E processing and perform no retry.
20	Perform modified retry processing. Call the failing utility, but do not compress the failing data set.

If any other value is returned, SMP/E converts it to 12, and command processing fails.

---

## Chapter 37. GIMDTS: Data Transformation Service Routine

Elements may have any of a variety of record formats, depending on how they are meant to be used. However, when elements are packaged inline in a SYSMOD, they must contain fixed-block 80 records. To help you package inline elements, SMP/E provides the GIMDTS service routine. GIMDTS is a background utility program that transforms data into fixed-block 80 records. For example, it can be used to format inline replacements for data elements. Although GIMDTS is packaged as part of SMP/E, it is a separate load module residing in SYS1.LINKLIB and runs independently from SMP/E.

The input for GIMDTS must meet these requirements:

- It must be a sequential data set or a member of a partitioned data set (PDS).
- It can contain either variable-length or fixed-length records.

The output from GIMDTS is in this format:

- It is a sequential data set or a member of a partitioned data set.
- It has these attributes:

```
RECFM=FB
LRECL=80
BLKSIZE=a multiple of 80
```

After using GIMDTS to transform an element into the required format, you can package the transformed data inline in a SYSMOD, following the associated data element MCS. Later, when the element is installed, it is changed back to its original format.

The following sections describe:

- Statements used to call GIMDTS
- Processing done by GIMDTS
- Return codes issued by GIMDTS

## Calling GIMDTS

Here are the JCL statements that are required to call GIMDTS:

```
//JOBx      JOB 'account','name',MSGLEVEL=(1,1)
//STEP1     EXEC PGM=GIMDTS
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD DSN=aaaaaaa,DISP=SHR
//SYSUT2    DD DSN=bbbbbbb,DISP=SHR
```

### EXEC

is the statement used to call GIMDTS.

### SYSPRINT

is used by GIMDTS for messages.

### SYSUT1

points to the sequential data set or PDS member containing the data to be transformed.

The record format (RECFM) must be F, FA, FM, FB, FBA, FBM, V, VA, VM, VB, VBA, or VBM. The input must not contain spanned records or MCSs.

### SYSUT2

points to the sequential data set or PDS member that will contain the transformed data. This data set must be on DASD.

The record format of this file should be fixed-block (RECFM=FB) with a BLKSIZE value that is a multiple of 80.

---

## Processing

GIMDTS first checks for the required data sets, opens the SYSUT1 and SYSUT2 data sets, and makes sure these data sets have the correct attributes. If necessary, GIMDTS changes the RECFM value of the SYSUT2 data set to FB and the BLKSIZE value to 3200. Next, GIMDTS reads records from the SYSUT1 data set and transforms the data.

After processing the data, GIMDTS writes the output records to the SYSUT2 data set. GIMDTS also issues messages to SYSPRINT indicating the input data set, the output data set, and the return code. If any errors occur during processing, GIMDTS issues messages describing the errors and stops processing. If an abend occurs, normal system abend processing is done.

---

## Return Codes

GIMDTS may issue the following return codes:

Return Code	Meaning
00	The input data was processed successfully.
04	The SYSUT2 data set was reblocked.
08	The input data was not processed successfully.
16	An I/O error occurred.



## Appendix A. SMP/E Naming Conventions

This appendix describes the naming conventions used by SMP/E. Table 36 summarizes the naming conventions you need to follow when using SMP/E. After this table are details on the naming conventions IBM follows for:

- HOLD reason IDs and classes
- Source IDs
- SYSMODs

<i>Table 36 (Page 1 of 3). Summary of SMP/E Naming Conventions</i>		
Entry or Value	Number of Characters	Other Requirements
DDDEF entry	1–8 alphanumeric (A–Z, 0–9) characters	Must match ddname of data set
DLIB zone or DLIBZONE entry	1–7 alphanumeric (A–Z, 0–9) or national (\$, #, @) characters	
Element entry	1–8 alphanumeric (A–Z, 0–9) characters	Naming convention for first character: A–I      Used by IBM J–Z      Available for users
FMID see SYSMOD ID		
FMIDSET entry	1–8 alphanumeric (A–Z, 0–9) characters	
Global zone or GLOBALZONE entry		Must be <i>GLOBAL</i>
Hold class	1–7 alphanumeric (A–Z, 0–9) characters	Naming convention for first character: A      Reserved for IBM use V      Reserved for user-assigned values  Values used by IBM: <ul style="list-style-type: none"> <li>• ERREL</li> <li>• UCLREL</li> </ul> For a description of specific hold classes used by IBM, see “Class Values” on page 808.
Hold reason ID: error	1–7 alphanumeric (A–Z, 0–9) characters	Number of the APAR used to report an error in a PTF  For details on the conventions IBM follows, see “Error Reason IDs” on page 807.

Table 36 (Page 2 of 3). Summary of SMP/E Naming Conventions

Entry or Value	Number of Characters	Other Requirements
Hold reason ID: system	1–7 alphanumeric (A–Z, 0–9) characters	<p>Naming convention for first character:</p> <p>A–U      Reserved for IBM use  V–Z      Reserved for user-assigned values</p> <p>Values used by IBM:</p> <ul style="list-style-type: none"> <li>• ACTION</li> <li>• AO</li> <li>• DELETE</li> <li>• DEP</li> <li>• DOC</li> <li>• EC</li> <li>• EXRF</li> <li>• FULLGEN</li> <li>• IOGEN</li> <li>• MSGSKEL</li> <li>• MVSCP</li> </ul> <p>For a description of specific system reason IDs used by IBM, see “System Reason IDs” on page 807.</p>
Hold reason ID: user	1–7 alphanumeric (A–Z, 0–9) characters	<p>To avoid conflicts with IBM reason IDs, follow the conventions for system reason IDs.</p> <p>For details on the conventions IBM follows, see “User Reason IDs” on page 808.</p>
OPTIONS entry	1–8 alphanumeric (A–Z, 0–9) characters	
Source ID	1–8 alphanumeric (A–Z, 0–9) characters	<p>Naming convention for first character:</p> <p>A–O      Reserved for user-assigned values  P–Z      Reserved for IBM use</p> <p>Values used by IBM:</p> <ul style="list-style-type: none"> <li>• PUTxxxx</li> <li>• SMCCOR</li> <li>• SMCREC</li> </ul> <p>For a description of the specific source IDs used by IBM, see “Naming Conventions for Source IDs” on page 808.</p>
SYSMOD ID	7 alphanumeric (A–Z, 0–9) characters	<p>Must start with a letter. Naming convention for first character:</p> <p>A–K      Used by IBM for functions  L–T      Available for users  U          Used by IBM for PTFs  V–Z      Used by IBM for APARs</p> <p>For details on the conventions IBM follows, see “Naming Conventions for SYSMODs” on page 809.</p>
Target zone or TARGETZONE entry	1–7 alphanumeric (A–Z, 0–9) or national (\$, #, @) characters	

Table 36 (Page 3 of 3). Summary of SMP/E Naming Conventions

Entry or Value	Number of Characters	Other Requirements
UTILITY entry	1–8 alphanumeric (A–Z, 0–9) characters	Must match the associated name specified in the appropriate OPTIONS entry  Should match the name of the associated utility program
ZONESET entry	1–8 alphanumeric (A–Z, 0–9) characters	Avoid using the same name as any of the target or DLIB zones defined in the global zone containing the ZONESET entry

## Naming Conventions for HOLD Reason IDs and HOLD Classes

The ++HOLD statement prevents SMP/E from installing a SYSMOD until some special action is taken. The type of action is indicated by the reason ID or class specified on the ++HOLD statement. A reason ID or class value can contain from 1 to 7 alphanumeric characters. To prevent conflicts between IBM- and user-specified values, there are naming conventions for the three types of HOLD reason IDs (error, system, and user), as well as for HOLD classes.

### Error Reason IDs

The reason ID for an error HOLD is the number of the APAR used to report an error in a PTF. Therefore, error reason IDs follow the naming conventions for APARs. These are described under “PTF, APAR, and USERMOD SYSMOD IDs” on page 810.

### System Reason IDs

The reason ID for a system HOLD is generally a brief indication of the kind of processing the SYSMOD requires. These are the general restrictions for These are the values currently used by IBM:

ID	Explanation
<b>ACTION</b>	The SYSMOD needs special handling before or during APPLY processing, ACCEPT processing, or both.
<b>AO</b>	The SYSMOD may require action to change automated operations procedures and associated data sets and user exits in products or in customer applications. The PTF cover letter describes any changes (such as to operator message text, operator command syntax, or expected actions for operator messages and commands) that can affect automation routines.
<b>DELETE</b>	The SYSMOD contains a ++DELETE MCS, which deletes a load module from the system.
<b>DEP</b>	The SYSMOD has a software dependency.
<b>DOC</b>	The SYSMOD has a documentation change that should be read before the SYSMOD is installed.
<b>EC</b>	The SYSMOD needs a related engineering change.

- EXRF** The SYSMOD must be installed in both the active and the alternative MVS/XA\* Extended Recovery Facility (XRF) systems at the same time to maintain system compatibility. (If you are not running XRF, you should bypass this reason ID.)
- FULLGEN** The SYSMOD needs a complete system or subsystem generation to take effect.
- IOGEN** The SYSMOD needs a system or subsystem I/O generation to take effect.
- MSGSKEL** This SYSMOD contains message changes that must be compiled for translated versions of the message changes to become operational on extended TSO consoles.
- If you want to use translated versions of the messages, you must run the message compiler once for the library containing the English message outlines, and once for each additional language you want to be available on your system. For details, see the *MVS/ESA Planning: Operations* manual.
- If you want to use **only** the English version of the messages, you do not need to run the message compiler. You should bypass this reason ID.
- MVSCP** The SYSMOD requires the MVS configuration program to be run for the change to take effect.

### User Reason IDs

The reason ID for a user HOLD is whatever you think is appropriate to describe why the SYSMOD should be held. Because IBM does not use these reason IDs, there are no restrictions on the values you can use. However, to prevent possible confusion with other IBM reason IDs, follow the naming conventions for system reason IDs.

### Class Values

A class value indicates an alternative way to release a held SYSMOD. These are the values currently used by IBM:

- | Class         | Explanation  |
|---------------|--|
| <b>ERREL</b>  | The SYSMOD is held for an error reason ID but should be installed anyway. IBM has determined that the problem the SYSMOD resolves is significantly more critical than the error reflected by the holding APAR. |
| <b>UCLREL</b> | UCLIN needed for the SYSMOD has been handled by IBM and no longer requires your attention.   |

---

### Naming Conventions for Source IDs

With the SOURCEID operand, you can associate a name (a source ID) with each SYSMOD processed by a single RECEIVE command, and then later process those SYSMODs as a group by using that source ID. The following values are currently used by IBM.

- PUTyynn** Each PTF is assigned a source ID in the format PUTyynn.
- *yy* is the year the PTF was made available as preventive service.
  - *nn* is a sequence number indicating the month that the PTF was made available as preventive service.

The PUTyynn source ID represents a monthly accumulation of PTFs that have been COR-closed (made available for corrective service) within a given month and assigned a RETAIN VOLID indicating this. The RETAIN VOLID reflects the monthly PUTyynn level no later than the 15th of the following month. For example, PUT9306 represents PTFs that were COR-closed in June and assigned a RETAIN VOLID of 9306 between July 1 and July 15.

- SMCCOR** This indicates a PTF that is approved for distribution, but has not been assigned a monthly PUTyynn source ID at the time your order was created. SMCCOR PTFs are not necessary for installation; however, they are made available for corrective service. If you plan to install any of the PTFs with this source ID, refer to the CORPE PSP bucket or latest available PUT bucket for the most current HOLD information. These PTFs are provided in case you experience the problem they fix and need to install them as corrective service.

- SMCREC** This indicates a PTF that is approved for distribution and is recommended for installation, but has not been assigned a monthly PUTyynn source ID at the time your order was created. These PTFs should be installed on your system. They were assigned a source ID of SMCREC for one of the following reasons:

- The PTF is needed to install a product.
- The PTF is needed to support new hardware.
- The PTF resolves an error for another PTF (PE-PTF).

To further enhance your ability to selectively install PUTyynn and SMCREC PTFs, the additional SOURCEIDs listed below are assigned to the applicable PTFs. (There will be PUTyynn and SMCREC PTFs that **do not** fit into the categories below and, therefore, are not assigned multiple source IDs.)

**HIPER** Identifies PTFs resolving a high-impact or pervasive APAR

**SPE** Identifies PTFs that are small programming enhancements

**PRP** Identifies PTFs that resolve PTFs in error

- SMPECNV** This is automatically added to SYSMOD entries when they are converted from SMP4 to SMP/E.

---

## Naming Conventions for SYSMODs

The specific naming conventions for a SYSMOD ID depend on the type of SYSMOD: function, PTF, APAR, or USERMOD. The following sections define IBM's naming conventions for SYSMOD IDs. This information is provided to help you develop a naming scheme for your own SYSMODs and avoid conflicts with IBM-written SYSMODs. However, these conventions are **not** requirements for

user-written SYSMODs, and SMP/E does not check whether a SYSMOD ID follows these conventions.

### Function SYSMOD IDs

The IBM convention for the SYSMOD ID of a function SYSMOD is *tccrrr*, where:

*t*

is the type of function. These are the values used by IBM:

- A Licensed vendor or business partner base function
- B Licensed vendor or business partner dependent function
- C, D Reserved for future use
- E Unlicensed base function
- F Unlicensed dependent function
- G Reserved for future use
- H Licensed base function
- I Reserved for future use
- J Licensed dependent function
- K Reserved for future use

Names of user-written functions can start with any letters other than these.

*ccc*

For functions provided by IBM, this is the product code. It must be three alphanumeric characters (no vowels).

*rrr*

For functions provided by IBM, this is the release number. It identifies a specific function within a product. These three characters identify a specific release of a product function and must be alphanumeric.

**Note:** The former convention for the SYSMOD ID of a function was *tvvrrr*.

### PTF, APAR, and USERMOD SYSMOD IDs

The IBM convention for the SYSMOD ID of a service SYSMOD or USERMOD is *tsnnnnn*, where:

*t*

identifies the type of SYSMOD. It is a single alphanumeric character. These are the values used by IBM:

- A–K Used by IBM for various levels of an APAR fix
- L–T Available for users
- U Used by IBM for PTFs
- V–Z Used by IBM for various levels of an APAR fix

*s*

is the system to which the SYSMOD and its associated product apply.

These are the values used by IBM for PTFs and APAR fixes:

- L, N, P Licensed products for MVS and VS1
- R Unlicensed products for MVS and VS1
- Y, Z MVS products only

Any valid character can be used for user SYSMODs.

*nnnn*

is an additional identifier for the SYSMOD. For PTFs and APAR fixes supplied by IBM, it is a number from 00001 to 99999.





---

## Appendix B. Sample SMP/E Cataloged Procedure

There are several ways to call SMP/E after it has been installed:

- Use the SMP/E dialogs.
- Submit a background job that calls GIMSMP, the program name for SMP/E. This job can call SMP/E either directly or in a cataloged procedure.

This section describes the types of information you need to provide if you use a cataloged procedure to invoke SMP/E. It discusses the following:

- Required JCL statements
- A sample cataloged procedure for SMP/E
- Formulas for estimating SMP/E's storage requirements

---

### Required JCL Statements

Unless you are using the SMP/E dialogs, you must provide the following JCL statements to invoke SMP/E:

- A JOB statement
- An EXEC statement
- DD (data definition) statements

### JOB Statement

The **JOB statement** describes your installation-dependent parameters. The JOB statement (or the EXEC statement, or both) can also include the REGION parameter to set the size of the region in which SMP/E runs. For details, see the *JCL User's Guide* or the *JCL Reference* manual.

**Note:** To enable the SMP/E job step to get the maximum space above the 16M line, you can specify REGION=0M. Or, if you prefer, you can specify a more specific region size. Formulas for estimating the region size for SMP/E are provided in "SMP/E Storage Requirements" on page 821.

### EXEC Statement

The **EXEC statement** must specify PGM=GIMSMP or the name of your cataloged procedure for calling SMP/E. (For an example of a cataloged procedure, see "Sample Cataloged Procedure for SMP/E" on page 815.) The following can be specified in the EXEC statement PARM parameter:

**CSI=dsname**

where *dsname* is the name of the CSI data set containing the global zone. (This data set is also known as the *master CSI*.) This parameter is used to enable SMP/E to allocate the master CSI data set dynamically.

**Note:** If there is an SMPCSI DD statement, the CSI=*dsname* operand is not allowed. If both are specified, SMP/E does **not** run.

**DATE=date**

where *date* can be one of the following:

U or IPL To use the IPL date of the system.

- REPLY To request the date from the operator. As a result, SMP/E issues message GIM399.
- yyddd* To specify a specific date, where *yy* is the year and *ddd* is the day of the year (the Julian date).

If **DATE** is not specified, the IPL date of the system is used.

### **LANGUAGE=xxx**

where *xxx* can be one of the following:

- ENU (US English)
- JPN (Japanese)

The **LANGUAGE** option defines which language to use for SMP/E messages.

**LANGUAGE** can also be specified as **L**. If **LANGUAGE** is not specified, the default is **LANGUAGE=ENU**.

You can specify **LANGUAGE=JPN** only if you have installed the Japanese language feature of SMP/E. If you have installed the Japanese language feature, you can specify **LANGUAGE=ENU** or **LANGUAGE=JPN**. (You do not need to install the English feature along with the Japanese feature.)

The output devices used must support the selected language and English single-byte characters. SMP/E does not check to verify that the output devices provide this support.

### **PROCESS=WAIT** or **PROCESS=END**

The **PROCESS** parameter is used to control how long a job should wait if a CSI or PTS data set is not immediately available because it is currently being used either by another job or by a dialog.

- **WAIT** causes the job to wait until the data set is available. A message is issued to the system operator every 30 minutes while the job is waiting.
- **END** causes the job to wait for 10 minutes. If the data set is still not available after the 10-minute wait, the command requiring the data set is stopped.

If **PROCESS** is not specified, the default is **PROCESS=WAIT**.

For more information on obtaining and sharing CSI data sets, see Appendix E, "Sharing SMP/E Data Sets."

Processing of the PTS data set is also affected by the **WAITFORDSN** value specified in its **DDDEF** entry. **WAITFORDSN** determines whether SMP/E should wait to allocate a data set that is not immediately available. If the **DDDEF** entry specifies **WAITFORDSN=NO** (or lets this value default to **NO**) and the data set is not available, allocation of the data set fails, regardless of the **PROCESS** value specified on the **EXEC** statement. If **WAITFORDSN=NO**, SMP/E does not wait to retry allocation of the data set.

For example, suppose a PTS with a disposition of **OLD** is already being used by a job, and a second job tries to access the same PTS data set by allocating it through a **DDDEF** entry. The **DDDEF** entry used by the second job for the PTS specifies **WAITFORDSN=NO**. As a result, allocation of the PTS fails for the second job.

## DD Statements

**DD statements** define the data sets that can be used in SMP/E processing. For information on the data sets required for each command, see the chapters on individual SMP/E commands.

**Note:** You can use DDDEF entries, rather than DD statements, to allocate many of the necessary data sets. For more information, see Appendix C.

---

## Sample Cataloged Procedure for SMP/E

Figure 137 on page 816 is a sample cataloged procedure, SMPPROC, that can be used to run SMP/E. The numbers to the left of the statements correspond to the notes that follow the example. When you write a cataloged procedure for SMP/E, remember the following:

- Tailor your own cataloged procedure to fit your system and processing requirements.
- You may want to use a single procedure for all SMP/E processing, or you may want to define multiple procedures for specific SMP/E commands and include in each one just those DD statements required for that command. For example, a special procedure for RECEIVE might include the SMPPTFIN DD statement but no DD statements for the target and distribution libraries.

**Note:** The SYSLIB concatenations for APPLY and ACCEPT should point to different libraries.

- Most of the data sets in the cataloged procedure can be allocated without DD statements. If you use the methods described for the data sets listed below, you may not need a cataloged procedure.
  - **Master CSI data set.** The master CSI data set can be specified on the CSI= parameter of the EXEC statement for GIMSMP, rather than on the SMPCSI DD statement. For more information about parameters you can specify on the EXEC statement, see “Required JCL Statements” on page 813.
  - **Target and distribution zones.** CSI data sets for target and distribution zones are normally dynamically allocated with zone indexes in the global zone. If you want to use the batch local shared resources (BLSR) subsystem, you must supply your own JCL statements. For an example of defining zone indexes, see “GLOBALZONE Entry (Global Zone)” on page 682. For more information about BLSR, see Appendix I, “Performance Improvements through Local Shared Resources (LSR)” on page 867.
  - **Other data sets.** Other data sets in the cataloged procedure can be defined with DDDEF entries. When you use DDDEF entries, only the data sets SMP/E needs for a particular run are allocated.

When you use DD statements, all the data sets defined must be online and allocated. Therefore, you might want to use a combination of DDDEF entries and a cataloged procedure shorter than the one in Figure 137. For more information about DDDEF entries, see “DDDEF Entry (Distribution, Target, and Global Zone)” on page 654.

- Although they are not shown in the sample cataloged procedure, the following DD statements may also be required:
  - An SMP\_CNTL DD statement, pointing to the commands that SMP/E processes, is required by all commands.
  - An SMPPTFIN DD statement, pointing to the source of the SYSMODs that are processed, is required by the RECEIVE command.
  - An SMPHOLD DD statement, pointing to the source of the HOLDDATA that is processed, is required by the RECEIVE command.
  - If the SYSMODs being processed were packaged in relative files, an SMPTLIB DD statement is required.
  - If any of the SYSMODs being installed contain elements packaged with the LKLIB operand, a DD statement for the ddname specified on that operand is required by the APPLY and ACCEPT commands.
  - If any of the SYSMODs being installed contain elements or JCLIN packaged with the TXLIB operand, a DD statement for the ddname specified on that operand is required by the APPLY and ACCEPT commands.
- If any of the required data sets (such as the SMPPTFIN) are not defined in the cataloged procedure or by DDDEF entries, they must be specified in the JCL used to call SMP/E.
- For more information about the data sets required by each SMP/E command, see the sections on data sets used, in the chapters describing the SMP/E commands.

```

//SMPPROC  PROC
//* ----- SYSOUT data sets -----
//SMPOUT  DD SYSOUT=A
//SMPRPT  DD SYSOUT=A
//SMPLIST DD SYSOUT=A
//SMPSNAP DD SYSOUT=A
//SMPDEBUG DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
1 //SMPPUNCH DD SYSOUT=B

//*----- SMP/E data sets -----
2 //SMPPTS  DD DSN=SYS1.SMPPTS,DISP=SHR
3 //SMPLTS  DD DSN=SYS1.SMPLTS,DISP=OLD
4 //SMPMTS  DD DSN=SYS1.SMPMTS,DISP=OLD
5 //SMPSTS  DD DSN=SYS1.SMPSTS,DISP=OLD
//SMPSCDS DD DSN=SYS1.SMPSCDS,DISP=OLD
//SMPLOG  DD DSN=SYS1.SMPLOG,DISP=MOD
//SMPLOGA DD DSN=SYS1.SMPLOGA,DISP=MOD
6 //* ----- Master CSI -----
//SMPCSI  DD DSN=SMPE.SMPCSI.CSI,DISP=SHR

```

Figure 137 (Part 1 of 2). Sample SMP/E Cataloged Procedure

```

7  /* ----- SMP/E temporary data sets-----
//SMPWRK1 DD UNIT=SYSDA,SPACE=(CYL,(2,1,5)),DISP=(,DELETE),
//          DCB=BLKSIZE=6160
//SMPWRK2 DD UNIT=SYSDA,SPACE=(CYL,(2,1,5)),DISP=(,DELETE),
//          DCB=BLKSIZE=6160
8  //SMPWRK3 DD DSN=data set name,
//          UNIT=SYSDA,SPACE=(CYL,(2,1,5)),DISP=(,CATALOG),
//          DCB=BLKSIZE=3120
//SMPWRK4 DD UNIT=SYSDA,SPACE=(CYL,(2,1,5)),DISP=(,DELETE),
//          DCB=BLKSIZE=3120
//SMPWRK6 DD UNIT=SYSDA,SPACE=(CYL,(2,1,5)),DISP=(,DELETE),
//          DCB=BLKSIZE=6160

/* ----- Utility data sets -----
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(2,1)),DISP=(,DELETE)
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(2,1)),DISP=(,DELETE)
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(2,1)),DISP=(,DELETE)
//SYSUT4 DD UNIT=SYSDA,SPACE=(TRK,(2,1)),DISP=(,DELETE)

/* ----- Assembler SYSLIB data set -----
9  //SYSLIB DD DSN=data set name,DISP=SHR
      . . .

/* ----- PARMLIB data set for JCLIN -----
//PARMLIB DD DSN=SYS1.PARMLIB,DISP=OLD

/* ----- Target libraries -----
//LINKLIB DD DSN=SYS1.LINKLIB,DISP=OLD
      . . .

/* ----- Distribution libraries -----
//AOSC5 DD DSN=SYS1.AOSC5,DISP=OLD
      . . .

//          PEND

```

Figure 137 (Part 2 of 2). Sample SMP/E Cataloged Procedure

- 1** **SMPPUNCH** is required for the GENERATE, REPORT, and UNLOAD commands. Because it may have a high level of output, SMPPUNCH should be directed to disk or tape.
- 2** **SMPPTS** should be coded with **DISP=SHR**. This allows concurrent jobs to share the PTS as much as possible. For more information on how SMP/E shares data sets, see Appendix E.
- 3** **SMPLTS** is required for changes to the base version of a load module.
- 4** **SMPMTS** is required for changes to macros that do not reside in a target library.
- 5** **SMPSTS** is required for changes to source code that does not reside in a target library.
- 6** **SMPCSI** DD statements should be coded with **DISP=SHR**. This allows SMP/E to share the CSI data sets as much as possible. If **DISP=OLD** is specified, no data-set sharing is attempted. For more information on how SMP/E shares data sets, see Appendix E.
- 7** **SMPWRK1—SMPWRK6** show only sample sizes for the data sets. The actual size required depends on the number of SYSMODs being processed and the number of elements within those SYSMODs.

**8 SMPWRK3** can be permanently allocated in order to reuse assemblies. For more information, see the description of the REUSE operand in Chapter 3.

**9 SYSLIB** concatenation depends on how you intend to use the distribution libraries. For details on which data sets to include and in what order, see the description after this figure.

If you use a different SYSLIB concatenation for APPLY and ACCEPT and prefer to use a SYSLIB DD statement, you should have at least two procedures. If you use DDDEFs to point to the different library concatenations, you can use one procedure. You can modify the examples to use the appropriate procedure.

### How to Determine the Appropriate SYSLIB Concatenation

There are two ways to determine the appropriate SYSLIB concatenation:

- The first approach treats the distribution libraries as backup libraries for SMP/E RESTORE processing.
- The second approach, normally used during system generation, treats the distribution libraries as totally separate from the target libraries.

**Note:** The examples shown are for processing a zone for SREL Z038 containing the MVS base control program (BCP). For other zones, follow the recommendations for the products residing in those zones.

#### Distribution Libraries as Backup Libraries for RESTORE

Treating the distribution libraries as backup libraries for SMP/E RESTORE processing ensures that the module accepted into the distribution libraries is exactly the same as the module applied to the target libraries. Thus, if you need to restore the module, the level returned to the system is exactly the same as had been previously installed. In this approach, the SYSLIB concatenation used during ACCEPT processing is the same as that used during APPLY processing. Figure 138 on page 819 is an example of the proper SYSLIB concatenation.

```

/* ----- Include SMPMTS first
/* ----- followed by all macro target libraries
/* ----- followed by all macro distribution
/*          libraries
//SYSLIB DD DSN=SYS1.SMPMTS,DISP=OLD
//          DD DSN=SYS1.MACLIB,DISP=OLD
//          DD DSN=SYS1.MODGEN,DISP=OLD
/* ----- IF YOU NEED TO ASSEMBLE JES SOURCE MODULES:
/* ----- either HASPSRC (JES2) or JES3MAC (JES3)
//          DD DSN=SYS1.HASPSRC,DISP=OLD
/* ----- SISTMAC1 if ACF/VTAM is installed
//          DD DSN=SYS1.SISTMAC1,DISP=OLD
//          DD DSN=SYS1.ATSOMAC,DISP=OLD
//          DD      .
//          DD      . other macro target libraries
//          DD      .
//          DD DSN=SYS1.AMACLIB,DISP=OLD
//          DD DSN=SYS1.AMODGEN,DISP=OLD
/* ----- IF YOU NEED TO ASSEMBLE JES SOURCE MODULES:
/* ----- either HASPSRC (JES2) or JES3MAC (JES3)
//          DD DSN=SYS1.HASPSRC,DISP=OLD
/* ----- AISTMAC1 if ACF/VTAM is installed
//          DD DSN=SYS1.AISTMAC1,DISP=OLD
//          DD      .
//          DD      . other macro distribution libraries
//          DD      .

```

Figure 138. APPLY and ACCEPT SYSLIB Concatenation

**Notes:**

1. This SYSLIB concatenation is the usual one for APPLY processing.
2. If you are using MVS/XA or MVS/ESA, the correct sequence is to place MACLIB before MODGEN. Placing MODGEN before MACLIB is appropriate only for MVS/370 users.

If you always accept the same SYSMODs you apply, when SMP/E does an assembly during ACCEPT processing, it uses the same level of macros as during APPLY processing. This makes sure the same object deck is generated.

**Risks for This Approach:** There are several risks in using this approach:

- You are accepting modules using levels of macros that may not be completely tested. If you later find an error in the macro, you have no way of redoing the assembly.
- If you apply SYSMODs that have macro changes between the APPLY and ACCEPT, the assembly done during ACCEPT processing may not be the same as processing done during APPLY processing.

Because macro changes are generally compatible, these risks are minimal.

### Distribution Libraries as Totally Separate from Target Libraries

Treating the distribution libraries as totally separate from the target libraries ensures that only the latest tested version of a macro is used during an assembly. Thus, the SYSLIB concatenation at ACCEPT is different from that at APPLY.

The SMPMTS data set contains macros from SYSMODs that are applied. Therefore, the proper SYSLIB concatenation for APPLY processing includes the SMPMTS data set, as is shown in Figure 139.

```
/* ----- Include SMPMTS first
/* ----- followed by all macro target libraries
/* ----- followed by all distribution
/*          libraries
//SYSLIB DD DSN=SYS1.SMPMTS,DISP=OLD
//          DD DSN=SYS1.MACLIB,DISP=OLD
//          DD DSN=SYS1.MODGEN,DISP=OLD
/* ----- IF YOU NEED TO ASSEMBLE JES SOURCE MODULES:
/* ----- either HASPSRC (JES2) or JES3MAC (JES3)
//          DD DSN=SYS1.HASPSRC,DISP=OLD
/* ----- SISTMAC1 if ACF/VTAM is installed
//          DD DSN=SYS1.SISTMAC1,DISP=OLD
//          DD DSN=SYS1.ATSOMAC,DISP=OLD
//          DD      .
//          DD      . other macro target libraries
//          DD      .
//          DD      .
//          DD      . any macro distribution libraries needed
//          DD      .
```

Figure 139. APPLY SYSLIB Concatenation: APPLY Different from ACCEPT

**Note:** If you are using MVS/XA or MVS/ESA, the correct sequence is to place MACLIB before MODGEN. Placing MODGEN before MACLIB is appropriate only for MVS/370 users.

During ACCEPT processing, the macros in the SMPMTS and in the target macro libraries are not considered to have been tested. The SMPMTS is, therefore, not concatenated. Figure 140 on page 821 shows the proper SYSLIB concatenation for ACCEPT.



```

/* ----- Include only macro distribution
/*      libraries
//      DD DSN=SYS1.AMACLIB,DISP=OLD
//SYSLIB DD DSN=SYS1.AMODGEN,DISP=OLD
/* ----- IF YOU NEED TO ASSEMBLE JES SOURCE MODULES:
/* ----- either HASPSRC (JES2) or JES3MAC (JES3)
//      DD DSN=SYS1.HASPSRC,DISP=OLD
/* ----- AISTMAC1 if ACF/VTAM is installed
//      DD DSN=SYS1.AISTMAC1,DISP=OLD
//      DD DSN=SYS1.ATSOMAC,DISP=OLD
//      DD      .
//      DD      . other macro distribution libraries
//      DD      .

```

Figure 140. ACCEPT SYSLIB Concatenation: APPLY Different from ACCEPT

**Risks for This Approach:** There is a risk with using this approach: The results of the assembly at ACCEPT may be different from those at APPLY. This can mean that if a SYSMOD is restored, and the distribution library module is installed into the target libraries, you may have a completely untested version of the module. This could introduce new errors into your system.

## SMP/E Storage Requirements

The REGION parameter on the JOB or EXEC statement for SMP/E must specify the amount of storage SMP/E needs. You can use the formula in Figure 141 to determine this amount.

```

1300K   (Size of SMP/E program)
+ 8K    (SMP/E fixed-size storage areas)
+ 20K   (reserved for system use ... OPEN,CLOSE)
+ 6 x   largest CSI control interval size
+ 2 x   largest PTS block size
+ 2 x   largest MTS/STS block size
+ 2 x   largest WRK1/WRK2/WRK3/WRK4/WRK6 block size
+ 1 x   largest LKLIB/TLIB block size
+ 36 x  largest PEMAX value
+ 1 x   largest size of programs invoked by SMP/E
+ 1 x   approximate storage for processing SYSMODs
        (the calculation below is needed only if you
        are running SMP/E on an MVS/370 system)
-----
TOTAL SIZE

```

Figure 141. SMP/E Virtual Storage Requirements (in Bytes)

To determine the amount of storage required by programs called by SMP/E, refer to the following manuals for your operating system:

- Link-edit utility and loader manual
- Assembler language reference manual
- Utilities manual
- Service aids manual

You can approximate the amount of storage SMP/E requires to process SYSMODs by multiplying the number of function SYSMODs to be processed by 120K and the number of service SYSMODs (PTFs, APARs, and USERMODs) to be processed by 0.75K. These storage estimates are based on the approximate sizes of work areas used during APPLY, RESTORE, and ACCEPT processing.

The maximum amount of storage that SMP/E attempts to obtain for internal entries is 4000K bytes. If your calculations show that the processing of a set of SYSMODs might exceed this amount, smaller subsets of that set should be processed.

---

## Appendix C. Dynamic Allocation

The processing of SMP/E commands requires a variety of data sets. You can either provide the DD statements for these data sets (such as in a cataloged procedure) or have SMP/E allocate the data sets dynamically. Dynamic allocation has the advantage that data sets are allocated only as they are needed; DD statements must successfully allocate all data sets, regardless of whether they are needed for the command being processed.

There are some drawbacks to using DD statements. For example, all the data sets defined by DD statements must be successfully allocated, regardless of whether they are needed for the command being processed. In addition, if you are running several SMP/E commands, you must be careful to use the correct DD statements for each command. If you are processing zones that are in different CSI data sets, you must make sure to provide a DD statement that points to each of those zones and their associated CSIs.

With dynamic allocation, you do not have these problems. Subsequent sections describe the sources from which SMP/E can get the information it needs to allocate data sets dynamically and how it chooses which of these sources to use.

---

### Sources of Information for Dynamic Allocation

SMP/E can use several sources of information to allocate data sets dynamically:

- DDDEF entries
- Module GIMMPDFT
- Standard defaults

### DDDEF Entries

You can use DDDEF entries to provide SMP/E with information it needs to allocate any of the following:

- Permanent data sets, such as target libraries, distribution libraries, and SMP/E data sets
- Temporary data sets
- SYSOUT data sets
- Work data sets
- Pathnames for elements and load modules residing in a hierarchical file system (HFS)

The name of the DDDEF entry must match the ddname of the data set it describes, and the entry must exist in the zone that uses the data set. DDDEF entries provide more flexibility than DD statements; they enable different zones to use different data sets for the same ddname, and they use resources more efficiently because they allow SMP/E to allocate only the data sets it needs.

DDDEF entries can include the following information:

- Data set name
- Unit type

- Volume serial number
- Initial data set status: NEW, OLD, MOD, or SHR
- Final data set status: KEEP, DELETE, or CATALOG
- How the data set is to be allocated: blocks, cylinders, or tracks
- Primary and secondary values for space allocation
- Whether the data set should be RACF-protected
- Whether the data set is SMS-managed
- Directory information used to allocate the pathname for an element or load module residing in a hierarchical file system (HFS)

For more information about DDDEF entries, see “DDDEF Entry (Distribution, Target, and Global Zone)” on page 654.

## Module GIMMPDFT

### Note

This section describes Diagnosis, Modification or Tuning Information. For a definition of Diagnosis, Modification or Tuning Information, see “Programming Interface Information” on page xxi.

Another way to provide SMP/E with information about data sets is through module GIMMPDFT, which is part of SMP/E. Unlike DDDEF entries, however, this information applies to all zones, not just to the set-to zone. (Although you can use GIMMPDFT to define temporary data sets, it is easier to tailor the definitions to your system if you use DDDEF entries instead.) CSECT GIMMPDFT in this module contains two tables:

- Table 1 defines SYSOUT data sets.
- Table 2 defines temporary data sets, such as the SMPWRKx data sets and the SYSUTx data sets.

Following Table 2 are six bytes that define the default space allocation for SMPTLIB data sets. When you get SMP/E, these two tables and the SMPTLIB information are set to zeros and blanks so they have no effect on SMP/E dynamic allocation.

You can change the values in GIMMPDFT to fit the needs of your system. SMP/E provides a sample USERMOD (SMP0001) containing superzap statements for entries in the tables. This USERMOD is in member GIMZPDFT in SYS1.SAMPLIB. You can use this SYSMOD as a model, change it to meet your needs, or install it as is.

The following sections describe the layout of the entries in GIMMPDFT and the sample entry values in USERMOD SMP0001.

**Note:** SMP/E does not strictly enforce rules for which subentries may be specified in DDDEF entries for specific data sets. To prevent allocation errors, refer to the JCL user's guide for your operating system.

**Table 1: SYSOUT Data Sets**

Table 1 in CSECT GIMMPDFT defines ddnames that may be allocated to the SYSOUT data set (for background processing) or to the terminal (for foreground processing). This table starts at offset hex 50 into CSECT GIMMPDFT and may contain up to 25 ten-byte entries. Table 37 shows the format of each entry.

Offset (DEC)	Offset (HEX)	Length	Description
+ 0	+ 0	8	ddname (padded to the right with blanks)
+ 8	+ 8	1	SYSOUT class: 00 = Do not allocate as SYSOUT. 40 = Use system default. (SYSOUT=*). Other values are treated as SYSOUT classes.
+ 9	+ 9	1	Terminal allocation: 5C = Allocate to terminal. Other values are ignored.

If you install sample USERMOD SMP0001 as is, Table 1 contains the values shown in Table 38.

Entry	DDNAME	SYSOUT
Entry 1	SMPOUT	SYSOUT=2 or terminal
Entry 2	SMPLIST	SYSOUT=2, not terminal
Entry 3	SMPRPT	SYSOUT=2, not terminal
Entry 4	SMPSNAP	SYSOUT=2, not terminal
Entry 5	SYSPRINT	SYSOUT=2, not terminal
Entry 6	SYSUDUMP	SYSOUT=2, not terminal
Entry 7	SMPPUNCH	SYSOUT=B, not terminal
Entry 8	SMPDEBUG	SYSOUT=2, not terminal
Entries 9—25	Not defined	

**Table 2: SMPWRKx and SYSUTx Data Sets**

Table 2 in CSECT GIMMPDFT defines ddnames that can be allocated for the SMPWRKx and SYSUTx data sets. This table starts at offset hex 14A into CSECT GIMMPDFT and can contain up to 25 entries of 65 bytes each. Table 39 shows the format of each entry.

Offset (DEC)	Offset (HEX)	Length	Description
+ 0	+ 0	8	ddname (padded to the right with blanks)

*Table 39 (Page 2 of 2). Table 2 in CSECT GIMMPDFT: Temporary Data Sets*

Offset (DEC)	Offset (HEX)	Length	Description
+ 8	+ 8	3	Allocation units: TRK = tracks CYL = cylinders Other values are treated as the BLKSIZE for blocks.
+11	+ B	3	Primary space (hex)
+14	+ E	3	Secondary space (hex)
+17	+11	3	Directory blocks (hex) (for partitioned data sets only)
+20	+14	8	Unit name (padded to the right with blanks)
+28	+1C	6	Volume serial number (padded to the right with blanks)
+34	+22	8	Data class name (padded to the right with blanks)
+42	+2A	8	Management class name (padded to the right with blanks)
+50	+32	8	Storage class name (padded to the right with blanks)
+58	+3A	7	DSNTYPE=LIBRARY or DSNTYPE=PDS (padded to the right with blanks)

If you install sample USERMOD SMP0001 as is, Table 2 contains the values shown in Table 40.

*Table 40. GIMMPDFT: Table 2 Values in SMP0001*

Entry	DDNAME	Space	Unit
Entry 1	SMPWRK1	(3120,(364,380,111))	SYSALLDA
Entry 2	SMPWRK2	(3120,(364,380,111))	SYSALLDA
Entry 3	SMPWRK3	(3120,(364,380,111))	SYSALLDA
Entry 4	SMPWRK4	(3120,(364,380,111))	SYSALLDA
Entry 5	SMPWRK6	(3120,(364,380,111))	SYSALLDA
Entry 6	SYSUT1	(3120,(380,760))	SYSALLDA
Entry 7	SYSUT2	(3120,(380,760))	SYSALLDA
Entry 8	SYSUT3	(3120,(380,760))	SYSALLDA
Entry 9	SYSUT4	(TRK,(1,1))	SYSALLDA
Entry 10	SYSPUNCH	(TRK,(25,10,10))	SYSALLDA
Entries 11—25	Not defined		

### Space Allocation for SMPTLIB Data Sets

The default space allocation values for SMPTLIB data sets start at offset hex 7A3 into CSECT GIMMPDFT and consist of three 2-byte entries. If you install sample USERMOD SMP0001 as is, these space allocation entries contains the values shown in Table 41.

Offset	Length	Description	Value
+ 0	2	Primary space allocation (in tracks)	0
+ 2	2	Secondary space allocation (in tracks)	0
+ 4	2	Directory blocks	0

### Standard Defaults

The SMPOUT and SYSPRINT data sets are critical for SMP/E to operate properly. Therefore, in case they are not defined, SMP/E has built-in defaults for them.

- SMPOUT is allocated either as SYSOUT (for background processing) or to the terminal (for foreground processing).
- SYSPRINT is allocated as SYSOUT.

## How Dynamic Allocation Works

Once SMP/E has determined which data sets are needed for the command it is processing, SMP/E checks whether DD statements have been provided for any of those data sets. SMP/E uses information from those DD statements in allocating the data sets to which they apply. If any data sets lack DD statements, SMP/E must allocate them dynamically. To get the information it needs to do this, SMP/E checks the following sources in the order shown.

1. **DDDEF entries.** If the zone specified on the SET command contains a DDDEF entry for the required data set, SMP/E uses that entry to allocate the data set. Otherwise, it checks the next source.
2. **Table 1 in GIMMPDFT.** If table 1 defines the data set, SMP/E uses that information to allocate the data set. Otherwise, it checks the next source.
3. **Table 2 in GIMMPDFT.** If table 2 defines the data set, SMP/E uses that information to allocate the data set. Otherwise, it checks the next source.
4. **Standard defaults.** If the data set is for SMPOUT or SYSPRINT, SMP/E uses the standard default to allocate the data set. Otherwise, data set allocation fails.

### Notes:

1. When a data set is part of a concatenation (such as the SYSLIB concatenation), SMP/E does not do the checking described above. All data sets in a concatenation must be defined the same way. For example, if a DDDEF entry specifies a concatenation, all the specified entries must also be defined by DDDEF entries.
2. For the cross-zone phase of APPLY and RESTORE processing, a DD statement cannot be used to allocate the SYSLIB for a cross-zone load module.

This library can be allocated only through a DDDEF entry in the cross-zone containing the LMOD entry for the cross-zone load module.

For each data set SMP/E tries to allocate (either dynamically or through DD statements), it writes information to the File Allocation report. For more information about this report, see “File Allocation Report” on page 463. SMP/E also writes the following information to the SMPLOG data set:

- Data set name or pathname
- Status
- Space allocation information
- Unit (only if specified in the DDDEF entry)
- Volume

In general, data sets that have been dynamically allocated remain allocated until the next SET command is processed. (Data sets allocated through Table 2 in GIMMPDFT remain allocated only until the next SMP/E command is processed, when they are freed and deleted.) If SMP/E fails to allocate a data set dynamically, or if SMP/E is requested to allocate a data set that is already allocated, it keeps a record of the error and does not try to reallocate the data set. When SMP/E processes the next SET command, it frees and deletes all dynamically allocated data sets and erases the records of allocation attempts that failed.

**Note:** If, in running a job containing several SET commands, you use DDDEF entries specifying SYSOUT for SMP/E output data sets (such as SMPOUT or SMPRPT), SMP/E produces multiple SYSOUT data sets. This can cause undesirable results; for example, the output may appear to be out of sequence from one SET command to the next. Therefore, when you run such a job, you may prefer to use DD statements instead of DDDEF entries for SMP/E output data sets.

For more information about the SET command and dynamic allocation, see “Examples” on page 350.



---

## Appendix D. Processing the SMP/E RC Operand

SMP/E keeps records of the highest return code for each command processed during a single SMP/E run. Before SMP/E processes a command, it checks whether the return codes for previous commands are less than the maximum allowed. If so, SMP/E can process the command; otherwise, the command fails.

You can change which return codes SMP/E checks for by specifying the RC operand on the command you want to process. This operand must be the last one specified on the command. It indicates which commands and return codes SMP/E is to check before processing the current command. The RC operand can specify any SMP/E command except DEBUG, REPORT, or SET. The return code must be a decimal number greater than or equal to 0 and less than 16. For example, if you want the ACCEPT command to run unless the return code from a previous APPLY command is higher than 8, and you want SMP/E to check the default values for JCLIN, UCLIN, and SET, you should code this RC operand on the ACCEPT command:

**RC (APPLY=08, JCLIN=08, UCLIN=08, SET=00)**

Table 42 on page 830 shows the default maximum return codes for each SMP/E command. Using these default values, for example, SMP/E processes an ACCEPT command if the return codes for previous commands meet these conditions:

- The highest return code for a SET command was 0. This condition is required because if the SET command fails, SMP/E does not know which zone to process for the subsequent commands.
- The highest return code for a JCLIN or UCLIN command was less than 8. This condition is required because these commands prepare data sets, such as the CSI, for processing by subsequent commands. If these commands fail, the other commands could either fail or run incorrectly.
- The highest return code for any other command was less than 12.

*Table 42. Default Maximum Return Codes for Commands Previously Processed*

Command to Be Processed	Maximum for All Commands Except JCLIN, UCLIN, and SET	Maximum for JCLIN or UCLIN	Maximum for SET
ACCEPT	12	08	00
APPLY	12	08	00
CLEANUP	12	08	00
CONVERT	12	08	00
DEBUG	Any	Any	Any
GENERATE	12	08	00
JCLIN	12	08	00
LINK	12	08	00
LIST	16	12	00
LOG	12	08	00
RECEIVE	12	08	00
REJECT	12	08	00
REPORT	Any	Any	Any
RESETRC	16	08	00
RESTORE	12	08	00
SET	Any	Any	Any
UCLIN	12	08	00
UNLOAD	12	08	00
ZONECOPY	12	08	00
ZONEDELETE	12	08	00
ZONEEDIT	12	08	00
ZONEEXPORT	12	08	00
ZONEIMPORT	12	08	00
ZONEMERGE	12	08	00
ZONERENAME	12	08	00

Return code processing for the RC operand is similar to processing for the default maximum return codes. Before SMP/E processes a command with the RC operand, it checks whether the return codes for previous commands are **less than or equal to** the maximums specified on the RC operand. If so, SMP/E can process the command; otherwise, the command fails. There is a difference, however. SMP/E checks return codes only for the commands specified on the RC operand. Return codes for any commands not specified do not affect processing for the current command. Therefore, if you use the RC operand, you should specify every command whose return code you want SMP/E to check.

---

## Appendix E. Sharing SMP/E Data Sets

Processing any SMP/E command requires a number of resources, such as shared and exclusive use of various data sets. These data sets include:

- The target libraries, distribution libraries, various temporary work data sets, and SYSOUT data sets
- All the CSI data sets and the SMPPTS data set

If you want to run more than one SMP/E job concurrently, either you must ensure data set integrity, or SMP/E must automatically provide that integrity.

You can manage the first category of data sets through:

- The DISP=OLD or DISP=SHR DD statement parameters. For more information, see Appendix B.
- DDDEF entries. For more information, see Appendix C or “DDDEF Entry (Distribution, Target, and Global Zone)” on page 654.
- The dynamic allocation tables in module GIMMPDFT. For more information, see Appendix C.

SMP/E itself controls how the CSI and SMPPTS data sets are shared for concurrent background jobs by:

- Using different types of access for different types of processing
- Dividing command processing into phases so each phase can use the correct type of access
- Using the system enqueue facility to obtain and release the data sets
- Providing special support for sharing the global zone and the SMPPTS

---

### Types of Access

Every SMP/E command needs access to the global zone. Some commands also require access to one or more other zones. In addition, commands may need different types of access. For example, LIST only reads data and can, therefore, share it with other LIST jobs. On the other hand, APPLY actually updates the target zone, and, therefore, needs exclusive use of that data set to ensure data set integrity.

To meet the needs of all the commands, SMP/E supports three types of access:

- **Read access with no control of the zone**

This access is required primarily during initialization for each command. For example, SMP/E might check the ZONEINDEX entries in the global zone to obtain the data set name for the target zone for the APPLY command. With this type of access, however, the data in the zone can be changed while SMP/E is checking it.

For this type of access no enqueue is issued. It is, therefore, called *read without enqueue*.

- **Read access with shared control of the zone**

This access is required during certain phases of processing to ensure that the data being checked is not being changed by another command. For example, when selecting SYSMODs to be applied, SMP/E must ensure that SYSMODs are not being received or rejected at the same time.

To gain this type of access, SMP/E issues a shared enqueue on the data set where the zone resides. This type of access is called *read with shared enqueue*.

- **Update access with exclusive control of the zone**

This access is required when the zone may be updated during command processing (as the target zone is during APPLY).

To gain this type of access, SMP/E issues an exclusive enqueue on the data set where the zone resides.

This type of access is called *update with exclusive enqueue*.

---

## Command Processing Phases

To avoid using the most restrictive type of access for the duration of a command, processing for each command is divided into phases. This way, SMP/E can obtain resources using the proper access type at the start of each phase, and free them at the end of each phase. This minimizes the amount of time a resource is tied up for exclusive use.

These are the important things to remember about command processing phases:

- Each command has at least two phases: initialization and termination. Most commands have one or more additional phases. For each command, the “Zone and Data Set Sharing Considerations” section of the chapter in this book devoted to that command describes the various phases of that command, which resources are required, and the type of access needed for that resource.
- During various phases of processing a command, SMP/E can use all three types of access for a given zone. For instance, during APPLY processing, the global zone is opened to obtain the name of the target zone CSI data set (read without enqueue), is then checked for which SYSMODs are eligible (read with shared enqueue), and is finally updated (update with exclusive enqueue).
- The most important use of command phases is to manage the global zone. This is important because each command must at least access the global zone to obtain further processing information, and most of the commands also update the global zone at some time during processing. Without command phases, almost all background SMP/E jobs would be serialized on the CSI data set containing the global zone.

---

## Using the System Enqueue Facility

SMP/E uses the system enqueue facility for most of its processing and provides its own support to control shared use of the global zone and SMPPTS.

At the start of each phase, SMP/E tries to obtain each required data set. It does this by issuing ENQ with the SYSTEMS parameter. This enables you to control access to the data sets across multiple systems. The name used for the ENQ is GIMSMP.*dsname*, where:

### **GIMSMP**

indicates that SMP/E issued the ENQ.

### *dsname*

is the name of the CSI data set containing the required zone.

You should use this information if you are using Global Resource Serialization (GRS) to ensure that the data set is not updated by multiple systems simultaneously.

If the required data sets are being used by another SMP/E job, the enqueue request fails, and SMP/E issues a dequeue request to free all data sets for which a successful enqueue was issued. It then retries the whole series of enqueue requests every 10 seconds until the resource is obtained. How long SMP/E continues this depends on the PROCESS value in the EXEC statement PARM field. You can specify either PROCESS=WAIT or PROCESS=END. If you do not specify PROCESS, the default is PROCESS=WAIT.

- **PROCESS=WAIT.** SMP/E waits for the required data set until it is obtained. Every 30 minutes, it sends a message to the system operator indicating that the job is still waiting for the data set.
- **PROCESS=END.** SMP/E waits 10 minutes for the data set. If the data set is still not available after that time, command processing ends with a return code of 12.

Once all data sets are available, that phase of the command can continue. At the completion of each phase, SMP/E issues a dequeue request for all data sets it requested and that are not required by the next phase of the command. SMP/E then starts the next command phase by requesting the additional data sets required for that phase.

**Note:** Because SMP/E manages zone sharing at the CSI data set level, it cannot process concurrent changes to different zones that are on the same CSI. The first job to run obtains exclusive use of the entire CSI data set. If you want to be able to process such concurrent changes, you should define the zones in different data sets. This is important to consider in determining how many CSI data sets you need and how to spread zones among them.

### Sharing the Global Zone and SMPPTS Data Set

Because of the way SMP/E controls access to the various CSI data sets, only one user can update a data set at a time. However, the global zone and SMPPTS data sets are affected whenever SYSMODs are installed; either the zone name must be placed in the global zone SYSMOD entries, or the SMPPTS MCS entries must be deleted. To avoid tying up these resources with requests for update access, SMP/E does the following:

- It records information in the zone being updated to reflect pending updates to the global zone and SMPPTS. Then, during one of the last phases in APPLY, ACCEPT, or RESTORE processing, SMP/E tries to make those pending changes to the global zone or SMPPTS. If either the global zone or SMPPTS is not available during this phase, SMP/E leaves the pending updates in the target zone or distribution zone.
- Whenever SMP/E opens a zone, the first thing it does is check for pending updates. If there are any, it tries to make the changes in the global zone and SMPPTS. If the global zone and SMPPTS are still not available, command processing continues, and the updates are left for the next access to that zone.
- Once the updates are made, SMP/E deletes the pending information from the target zone or distribution zone.

This processing allows multiple jobs to run concurrently against a common global zone and SMPPTS data set while protecting those data sets from concurrent updates and ensuring that all pending updates are made.

---

## Appendix F. Restricting Which Utilities Are Called

### Note

This chapter describes Diagnosis, Modification or Tuning Information. For a definition of Diagnosis, Modification or Tuning Information, see “Programming Interface Information” on page xxi.

When you install SYSMODs, SMP/E calls system utility programs to make the changes or to recover from failures. For security, you may want to restrict the programs SMP/E can call. To do this, you can change module GIMUTTBL, which lists all the utility programs SMP/E should consider valid.

---

### Summary of GIMUTTBL Processing

When SMP/E is installed, GIMUTTBL is copied from SYS1.AOS12 to SYS1.LINKLIB. SMP/E uses the resulting load module in SYS1.LINKLIB to choose the programs it can call. When SMP/E processes a call for a particular utility program, it follows these steps to determine whether the program can be called:

1. Before SMP/E calls the utility program, it loads module GIMUTTBL.
2. SMP/E checks the list of valid program names in GIMUTTBL.
  - If the list is empty, SMP/E assumes that any utility program is valid. So, if you do not change GIMUTTBL, SMP/E does not check for valid programs.
  - If the list has entries, SMP/E checks the list for the name of the program to be loaded.
    - If SMP/E finds the name, processing continues.
    - If SMP/E does not find the name, command processing fails.
3. If SMP/E does not find GIMUTTBL, command processing fails. This happens if you delete GIMUTTBL from SYS1.LINKLIB after installing SMP/E.

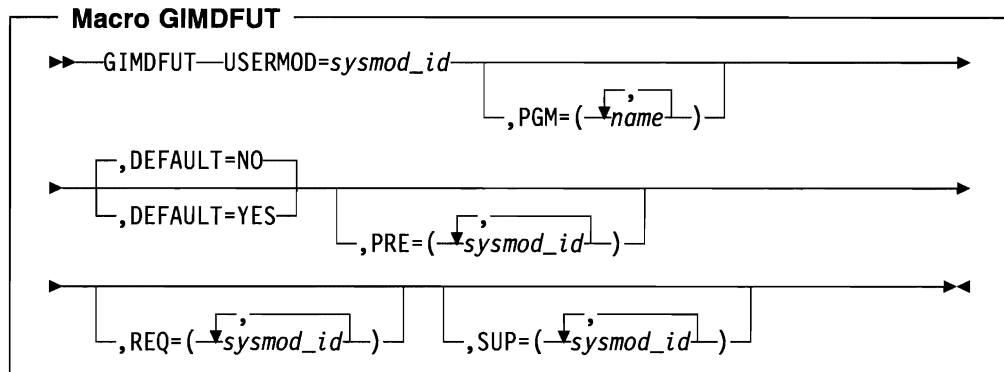
---

### Using GIMUTTBL

If you want to use module GIMUTTBL to restrict which utility programs SMP/E can call, follow these steps:

1. Code macro GIMDFUT with the parameters needed to make the desired changes. This macro comes with SMP/E.
2. Assemble macro GIMDFUT. This creates a USERMOD that lists the valid program names and replaces GIMUTTBL.
3. Use RECEIVE, APPLY, and ACCEPT to install the USERMOD for GIMUTTBL.

These are the parameters you can specify on macro GIMDFUT:



**USERMOD**

defines the SYSMOD ID of the USERMOD SMP/E creates.

**PGM**

is the list of program names SMP/E can invoke. Specify any number of names enclosed in parentheses and separated by a comma. Each value must contain from 1 to 8 alphanumeric characters.

If you specify YES for the DEFAULT operand, you can omit the PGM operand. Otherwise, you must define it.

**DEFAULT**

indicates whether the names of standard utility programs should be included in GIMUTTBL. These names are added to any names defined with the PGM operand. The default value is NO.

If you specify YES for DEFAULT, the following standard utility programs are included in GIMUTTBL:

- BPXCOPY** Hierarchical file system copy utility
- IEV90** Assembler utility
- IDCAMS** Access method services utility
- IEBCOPY** Compress utility, copy utility, retry utility
- IEBUPDTE** Update utility
- IEWBLINK** Binder link-edit utility
- IEWL** Link-edit utility
- IMASPZAP** Superzap utility

**PRE**

lists the prerequisite SYSMODs that should be specified on the ++VER statement of the USERMOD. Specify any number of SYSMOD IDs; enclose them in parentheses, and use a comma as a separator after every one except the last. Each SYSMOD ID must contain 7 alphanumeric characters.

**REQ**

lists the requisite SYSMODs that should be specified on the ++VER statement of the USERMOD. Specify any number of SYSMOD IDs; enclose them in parentheses, and use a comma as a separator after every one except the last. Each SYSMOD ID must contain 7 alphanumeric characters.



**SUP**

lists any superseded SYSMODs that should be specified on the ++VER statement of the USERMOD. Specify any number of SYSMOD IDs; enclose them in parentheses, and use a comma as a separator after every one except the last. Each SYSMOD ID must contain 7 alphanumeric characters.

Figure 142 is an example of macro GIMDFUT and the resulting USERMOD.

```

-----+-----1--...-----10-----+-----11-----+---
                                GIMDFUT  USERMOD=USER003,           X
                                    PGM=(MYCOPY,IFOX00,             X
                                    MYRETRY,USERZAP),               X
                                    DEFAULT=YES,                    X
                                    SUP=(USER001,USER002)
                                    END.

This would generate the following user modification:

++USERMOD(USER003).
++VER(Z038) FMID(HMP1802)
    SUP(
        USER001
        USER002
    ).
++MOD(GIMUTTBL) DISTLIB(AOS12).
...
... object code for GIMUTTBL
...

```

Figure 142. Sample USERMOD Created by GIMDFUT

After this sample USERMOD is installed, module GIMUTTBL contains entries for the following utility programs:

- From the user modification:
  - MYCOPY
  - IFOX00
  - MYRETRY
  - USERZAP
- From the GIMDFUT defaults:
  - BPXCOPY
  - IEV90
  - IDCAMS
  - IEBCOPY
  - IEBUPDTE
  - IEWBLINK
  - IEWL
  - IMASPZAP



## Appendix G. Summary of All SMP/E Releases

SMP/E contains new functions, in addition to those provided by SMP4. Table 43 summarizes these changes release by release. A brief explanation of these changes follows the table.

Table 43 (Page 1 of 3). SMP/E Release Summary

Change	Release Where Available			
	R6 or Earlier	R7	R8	R8.1
ADDIN support deleted	X	X	X	X
Additional element types	X	X	X	X
APAR regression reporting	X	X	X	X
Automatic rereceive of reworked SYSMODs	X	X	X	X
BLOCKSIZE=8800 for SMP/E data sets				X
CBIPO dialogs	X	X	X	X
CBIPO dialogs: BLOCKSIZE=0 support		X	X	X
CBIPO dialogs: Default space allocation for JCL data set (improved method)		X	X	X
CBIPO dialogs: Format of MLPA member		X	X	X
CBIPO dialogs: FunctionPac* support		X	X	X
CBIPO dialogs: Job management function		X	X	X
CBIPO dialogs: Dynamic dump allocation				X
CBIPO dialogs: MVSCP elimination				X
CBIPO dialogs: New Table Format				X
CBIPO dialogs: Symbolic substitution in data set names				X
CBIPO dialogs: 4-digit device addresses				X
Cleanup of SMP/E data sets related to target zones	X	X	X	X
Comparing the SYSMODs installed in two zones	X	X	X	X
Concatenation of up to 123 DDDEF entries	X	X	X	X
Copy processing: Suppression of member names			X	X
Creating and updating DDDEF entries through the SMP/E dialogs		X	X	X
Cross-product load modules: Implicitly including modules (link-edit automatic library call support)			X	X
Cross-product load modules for products in different zones: LINK command		X	X	X
Cross-product load modules for products in the same zone: improved processing		X	X	X
Cross-zone requisite reporting	X	X	X	X
CSECT delete support	X	X	X	X
Data element changes	X	X	X	X
Data Facility Product (DFP) support		X	X	X

Table 43 (Page 2 of 3). SMP/E Release Summary

Change	Release Where Available			
	R6 or Earlier	R7	R8	R8.1
Data set sharing	X	X	X	X
Deleting and superseding the same function	X	X	X	X
Dialogs	X	X	X	X
Dumps for VSAM errors	X	X	X	X
Dynamic data set allocation	X	X	X	X
Exception SYSMOD processing	X	X	X	X
Exception SYSMOD reporting	X	X	X	X
Expanded Service Option (ESO): support for			X	X
Format change for CSI and SCDS	X	X	X	X
GENERATE command	X	X	X	X
GENERATE command: REPLACE option	X	X	X	X
GROUP processing improvements	X	X	X	X
Hierarchical file systems			X	X
ISPF Version 2 Release 3 and ISPF/PDF Version 2 Release 3 required	X	X	X	X
Japanese messages and dialogs	X	X	X	X
JCLIN reports	X	X	X	X
JCLIN saved at ACCEPT time	X	X	X	X
Messages rewritten	X	X	X	X
MOVE/RENAME/DELETE processing	X	X	X	X
MVS/370 support: SMP/E Release 8.1 is last release supporting				X
New packaging	X	X	X	X
Obsolete MCS type and operands deleted	X	X	X	X
Out-of-space errors: Improved recovery			X	X
Panel identifier for dialogs	X	X	X	X
Performance improvement: Automatically using batch local shared resources (BLSR)	X	X	X	X
Performance improvement: Using batch local shared resources (BLSR) subsystem with expanded storage hiperspaces	X	X	X	X
RACF protection of dynamically allocated data sets	X	X	X	X
RACF to permit read-only access to CSI data sets	X	X	X	X
Regressed subentries deleted	X	X	X	X
REJECT improvements	X	X	X	X
RELFILES: High-level qualifier for			X	X
RELFILES: Receiving from DASD			X	X
REPORT CALLLIBS command			X	X
Root cause analysis	X	X	X	X

Table 43 (Page 3 of 3). SMP/E Release Summary

Change	Release Where Available			
	R6 or Earlier	R7	R8	R8.1
SMPLOG improvements	X	X	X	X
SMPTLIB improvements	X	X	X	X
SMPWRK5 deleted	X	X	X	X
SMPWRK6 required	X	X	X	X
Source ID changes	X	X	X	X
Source ID exclusion	X	X	X	X
Source ID report	X	X	X	X
SYSGEN no longer supported for SMP/E	X	X	X	X
SYSMOD selection	X	X	X	X
SYSMOD selection: Inline source identifier	X	X	X	X
SYSMOD selection: RECEIVE by FMID	X	X	X	X
SYSMOD Status report accuracy	X	X	X	X
UPGRADE operand deleted	X	X	X	X
Utility error isolation	X	X	X	X
Virtual storage constraint relief	X	X	X	X
VSAM data sets	X	X	X	X
VS1 no longer supported	X	X	X	X
Zone descriptions	X	X	X	X
Zone editing	X	X	X	X
Zone utility commands	X	X	X	X
ZONESETs with mixed zones	X	X	X	X

## ADDIN Support Deleted

The UNLOAD command no longer processes ADDIN control statements. The ADDINPUT operand has been deleted from the UNLOAD command, and the SMPADDIN data set is no longer supported.

## Additional Element Types

In addition to macros, modules, and source, SMP/E now recognizes other types of data as program elements. For example, CLISTs no longer have to be packaged as macros; they can be packaged as CLISTs. These new elements, generally called *data elements*, can contain either variable-length or fixed-length records, and fixed-length records can have a logical record length other than 80. The following changes have been made to support these new data elements:

- New modification control statements (MCSs) to add or replace data elements
- New data element entries in the CSI to track changes to data elements

- New LIST, UCLIN, and UNLOAD operands for data element entries

If data elements are packaged in relative files, they can remain in their original format. However, if they are packaged inline, they must be in fixed-block-80 format. To help package elements not in this format, SMP/E provides a new service routine, GIMDTS, which transforms elements from their original format into the required format. These transformed elements can then be packaged inline. When SMP/E installs these transformed elements, it transforms them back to their original format.

---

### APAR Regression Reporting

When SMP/E installs a function SYSMOD, it issues a warning message if the function can potentially overlay (regress) an APAR fix that had been previously installed.

---

### Automatic Rereceive of Reworked SYSMODs

Sometimes minor changes are made in a SYSMOD that has already been received into the SMPPTS. SMP/E automatically rereceives this SYSMOD when it receives other SYSMODs from the same source, as long as the reworked SYSMOD specifies the REWORK operand and is more recent than the version already on the system. This eliminates the need to delete the old version of the SYSMOD to explicitly receive the new one.

---

### BLOCKSIZE=0 Support (CBIPO Dialogs)

For systems with DFP Version 3, the CBIPO dialogs now support the BLOCKSIZE=0 parameter for data set allocation, which is used by DFP to allocate the data set with an optimum block size.

---

### BLOCKSIZE=8800 for SMP/E Data Sets

For the RELFILE data sets, target libraries, and distribution libraries containing SMP/E Release 8.1, data sets that are allocated with RECFM=FB and LRECL=80 are now allocated with BLKSIZE=8800.

---

### CBIPO Dialogs

SMP/E now includes dialogs that can be used to install CBIPO orders. These dialogs can be used instead of the existing batch job installation process.

With the CBIPO dialogs, you can:

- Install a system or subsystem from a CBIPO or redistribution tape.
- Reinstall existing user data while installing a new system or subsystem from a CBIPO or redistribution tape.
- Redistribute a system or subsystem by using either the *local* or *remote* redistribution method.

For both types of redistribution, the data set, DASD, and/or catalog configuration of the copied system or subsystem need not be the same as the configuration of the original system or subsystem.

- Process I/O definitions for an MVS system that was installed using the CBIPO dialogs:
  - Process an MVSCP deck
  - Include existing I/O definition files (IODFs) for use during IPL and to be copied during reinstall or redistribution processing
- Connect a previously installed subsystem to another MVS system.

---

## Cleanup of Target Zone Data Sets

The CLEANUP command deletes entries from target zone data sets used to temporarily store macros, source, and copies of target zone entries. SMP/E automatically deletes these entries—for example, when a SYSMOD installed in the target libraries is later installed in the distribution libraries. However, there are times when these entries are not deleted:

- If a SYSMOD is installed in several target libraries created from the same distribution library.

When it installs the SYSMOD in the distribution library, SMP/E deletes entries only from the data sets for the target zone that the distribution zone names as its related zone.

- If a SYSMOD is installed in the distribution libraries before it is installed in the target libraries.

In both of these cases, the CLEANUP command can be used to delete the entries from the data sets, eliminating the need to reinstall the SYSMOD in the distribution library each time entries in the data sets for a target library must be deleted. This saves time and avoids the risk of using up space in the distribution library.

---

## Comparing the SYSMODs Installed in Two Zones

The REPORT command can now be used to compare the SYSMOD content of two zones. Specifically, the REPORT SYSMODS command tells you which SYSMODs (that have been installed in one zone) are applicable to a second zone but not yet installed in it.

---

## Concatenated DDDEF Entries

SMP/E now allows you to concatenate up to 123 DDDEF entries. However, the actual number of partitioned data sets that can be concatenated depends on the operating system you are running under. To determine the maximum number of data sets you can concatenate, see the data management manual for your operating system.

---

## Creating and Updating DDDEF Entries through the SMP/E Dialogs

The panel flow for creating and updating DDDEF entries has been simplified:

- When a new DDDEF entry is being modeled after an existing entry, the dialogs display the appropriate DDDEF entry panel, primed with values from the entry being used as a model.
- When an existing DDDEF entry is selected, the dialogs determine the type of DDDEF entry and display the appropriate DDDEF entry panel.

---

## Cross-Product Load Modules for Products in Different Zones: LINK Command

The new LINK command allows you to link-edit specific modules from one zone into load modules controlled by a different zone. This postinstallation support is helpful for a product that must do either of the following:

- Include in one of its own load modules one or more specific modules from another product that is controlled by a different zone
- Include one or more of its own modules in a load module that is controlled by a different zone

Because LINK processing is under SMP/E control, it does the following:

- Reduces the need for preinstallation and postinstallation steps that must be run outside of SMP/E to accomplish the same task
- Tracks the existence of modules that are part of load modules in other zones, so subsequent APPLY and RESTORE processing can automatically maintain the affected load modules

---

## Cross-Product Load Modules for Products in the Same Zone: Improved Processing

Improved load module management reduces the amount of postinstallation processing needed to manage products whose load modules include modules from other products defined in the same target zone. This improvement is helpful when the product that provided the “borrowed” modules is deleted by a new version or release.

- In the past, when such products were deleted, SMP/E did not keep a record of the modules borrowed from the deleted product. As a result, when the new version or release of the product reintroduced the borrowed modules, SMP/E had no way of knowing that these modules should be included in another product's load module. Instead, UCLIN had to be run, and often link-edits had to be done outside of SMP/E.
- Improved load module management eliminates the need for this extra work. SMP/E maintains a record of any modules from a deleted product that were included in a load module of another product. If the deleted modules are reintroduced, SMP/E automatically link-edits the load module to include the borrowed modules.



---

## Cross-Zone Requisite Reporting

SMP/E helps keep related products in different zones at the same service level by reporting on conditional requisites affecting functions in a user-defined set of zones.

Conditional requisites indicate dependencies between functions. For example, a SYSMOD to correct an interface error in one function may require a second SYSMOD to change another function that uses the interface. The SYSMOD that corrects the interface error names the other SYSMOD as a conditional requisite—that is, a change that is needed if the second function is installed.

In previous releases, when SMP/E installed SYSMODs naming conditional requisites, it checked the zone being processed for functions that needed these requisites. SMP/E now extends this checking to a group of zones, or ZONESET. This ZONESET is used with the new REPORT command to list which zones need which requisites and to write the SMP/E commands that can be used to install the changes.

---

## CSECT Delete Support

SMP/E can delete CSECTs from load modules containing modules from different products. This allows the load module to be restructured when one of the functions is applied, without having to process link-edit steps before running SMP/E or doing a system generation after running SMP/E.

---

## Data Element Changes

SMP/E now recognizes the following new MCS types for data elements:

++BSINDxxx	Index for online publications library, or bookshelf
++CGMxxx	Graphics source for an online book
++DATA6xxx	For IBM use only, to support translated versions of elements not covered by any existing data types
++EXEC	EXEC
++FONTxxx	Printer Font Object Contents Architecture (FOCA) font
++IMGxxx	Graphics image for an online book
++PSEGxxx	Graphics page segment for an online book

In addition, SMP/E now recognizes ENP as the language abbreviation for upper-case English.

---

## Data Facility Product (DFP) Support

SMP/E now supports the following Data Facility Product (DFP) functions:

- Storage Management Subsystem (SMS) classes on MVS/XA and MVS/ESA systems

Data classes, storage classes, and management classes can now be specified in SMP/E DDDEF entries to allocate new SMS-managed data sets dynamically.

SMP/E data sets that can benefit from this new support include SMPTLIB and SMPWRKx data sets.

- A new type of partitioned data set called *partitioned data set extended* (PDSE)  
SMP/E recognizes the DSNTYPE field, which is used to identify a data set as a partitioned data set (PDS) or a PDSE. In addition, SMP/E supports PDS or PDSE members with more than 16 aliases.
- Load modules contained in PDSE program libraries  
SMP/E supports load modules residing in PDSEs whose external names or alias names do not exceed eight characters.  
In addition, SMP/E supports the new binder and loader program, IEWBLINK, which is used to create load modules. SMP/E also recognizes new options that can be specified when invoking the binder.

---

## Data Set Sharing

To ensure the integrity of the CSI and SMPPTS data sets during SMP/E command processing, especially when several SMP/E commands are processed concurrently, SMP/E automatically accesses these data sets. Depending on the needs of the command being processed, SMP/E allows the following types of access:

- **Read access:** to obtain information without making any changes.
- **Read access with shared control:** to obtain information while ensuring that data is not being changed by another command.
- **Update access with exclusive control:** to change information while ensuring that data is not being changed by another command.

---

## Default Space Allocation for JCL Data Set: Improved Method

The CBIPO dialogs now use an improved method to determine the default space allocation to be used for the data set containing the generated JCL.

---

## Deleting and Superseding the Same Function

A function SYSMOD can now both delete and supersede another function SYSMOD. This way, SMP/E cleans up entries for the deleted function and recognizes that the new function satisfies the requisites specified for the deleted function.

---

## Dialogs

You can use SMP/E by running batch jobs or by using the SMP/E dialogs, which run as an application under ISPF/PDF. With the SMP/E dialogs, you can do the following:

- Define the contents of the system
- Install products and service from a variety of sources by following a guided step-by-step process
- Display SMP/E records containing information about the contents and status of the system

- Run individual SMP/E commands
- Learn how to use the SMP/E dialogs
- Obtain guidance for using specific SMP/E dialog panels

The SMP/E dialogs have the following usability attributes:

- The panels present information in a consistent format.
- The dialogs are designed to be easily understood by occasional or less experienced SMP/E users. The panels display the options, selections, or responses you can make. For further assistance, these are backed up by tutorial panels.
- SMP/E dialogs reduce the amount of data required to perform a task because:
  - When appropriate, user input for one processing run is saved and used during the next run.
  - Information saved in ISPF tables or SMP/E data sets reduces the amount of data you have to enter.
  - In many dialogs, defaults or installation-defined values are assigned, so you need to enter the data only if the default is not wanted.
- The dialogs simplify the performance of certain tasks by handling SMP/E syntax and other technical details for you. The dialogs alert you when any overt action (such as the submission of a background job) is to take place, and you have the option of editing the job stream, submitting the job, or deferring or canceling the job.
- A dialog session can be suspended and resumed during the current logon session or during another logon session. You have the option of resuming the dialog at the point at which it was suspended, or beginning a new dialog invocation.

---

## Dumps for VSAM Errors

The DEBUG command can now be used to request a dump of the VSAM RPL control block when an error occurs during an attempt to access a CSI data set. This RPL dump can be used to analyze the cause of the error.

In the past, SMP/E automatically dumped the VSAM RPL only when the return code was 8 or above. Now the user can request the dump to be taken when certain specific types of internal SMP/E processing are being done and the return code is 8 or above. This allows you to determine the cause of the error more closely. To further aid problem determination, the RPL dump has been expanded to include additional RPL information.

---

## Dynamic Data Set Allocation

SMP/E can dynamically allocate each data set needed to process an SMP/E command using information previously stored in the zone being processed. SMP/E dynamic data set allocation can be overridden by supplying data definition statements (DD statements) when SMP/E is called.

### Dynamic Dump Allocation (CBIPO Dialogs)

The PARMLIB/PROCLIB portion of the Tailor options of the CBIPO dialogs facilitates the exploitation of the dynamic dump allocation support provided in MVS/ESA SP Version 5. If the dialogs are processing an order containing MVS/ESA SP Version 5, the user can specify where the dump information is to be written (via SMS classes or a group of DASD volumes), how the dump data sets are to be named, and whether automatic allocation of dump data sets is to be enabled. The dialogs use this information to create the necessary DUMPDS commands in the user-specified COMMNDxx PARMLIB member.

**Note:** The dialogs continue to support preallocated dump data sets. In addition, dynamically allocated dump data sets can coexist with preallocated dump data sets.

Configurations that share resources, such as a master catalog, should make use of this support instead of preallocating dump data sets. Dynamic dump allocation can help ensure that the names of dump data sets sharing a master catalog are unique within a parallel sysplex.

---

### Exception SYSMOD Control

To help you manage SYSMODs requiring special processing, SMP/E supports ++HOLD and ++RELEASE statements. ++HOLD statements prevent SMP/E from installing specified SYSMODs until some special action is taken. When ++HOLD statements are processed by the RECEIVE command, the specified SYSMODs are held and cannot be installed until the necessary processing has been done. ++RELEASE statements remove SYSMODs from hold status without resolving the reason for which they are being held. These are the reasons for which a SYSMOD can be held:

- **ERROR:** An APAR reported an error in the SYSMOD. The SYSMOD should not be processed until the APAR is resolved. A PTF held for this reason is also called a program error PTF, or PE-PTF. SMP/E automatically releases the SYSMOD when a SYSMOD that supersedes the APAR is processed.

The tape for preventive service has a file containing error HOLD statements for SMP/E processing.

- **SYSTEM:** Special action outside normal SMP/E processing is required for the SYSMOD. Examples are SYSMODs requiring a SYSGEN after they are installed, or SYSMODs that require the installation of an associated engineering change (EC) level. System HOLD statements can appear in the SYSMOD itself or in a separate HOLD input file.

Two new system hold reason IDs can now be specified for SYSMODs:

- AO is used for a SYSMOD that may require action to change automated operation procedures and associated data sets and user exits in products or in customer applications.
- MSGSKEL is used for a SYSMOD containing message changes that must be compiled for translated versions of message changes to become operational on extended TSO consoles.
- **USER:** The SYSMOD requires special processing because of a decision you have made.

In addition to an exception status, a HOLD statement can also indicate a class, an alternative reason for releasing a held SYSMOD. For example, a SYSMOD may contain a system hold for UCLIN processing and a class hold for SYSGEN processing. This means that the UCLIN processing is not necessary if you plan to do a SYSGEN after installing the SYSMOD. Or a group of exception SYSMODs may contain different holds but specify the same class. If that class condition is met, the entire group of exception SYSMODs is released for processing. Thus, a class makes exception SYSMOD processing more flexible.

Formerly, a SYSMOD could be prevented from being installed because it was superseded by a SYSMOD that was being held. SMP/E can now process a lower-level SYSMOD if it finds that the superseding SYSMOD is in HOLD status. That way, the system stays at the highest possible service level.

---

## Exception SYSMOD Reporting

The REPORT command can now be used to determine whether any SYSMODs that have already been installed are now exception SYSMODs. Specifically, the REPORT ERRSYSMODS command lists SYSMODs after whose installation ++HOLD statements were received and whose HOLDERROR reason IDs have not yet been resolved.

---

## Expanded Service Option (ESO) Support

Where available, the Expanded Service Option (ESO) has replaced the program update tape (PUT) process as the vehicle for delivering preventive service. SMP/E has replaced its support for PUTs (including the format of PUT tapes) with support for ESO tapes.

---

## Format Change for CSI and SCDS Data Sets

The format of the records in CSI and SCDS data sets for SMP/E Release 8.1 is different from the format for SMP/E Release 4, or earlier. Any existing CSI or SCDS data sets used with SMP/E Release 4, or earlier, that are used with SMP/E Release 8.1 must, therefore, be converted to SMP/E Release 8.1 format. The SMP/E CONVERT command has been updated to do the conversion.

**Note:** CSI and SCDS data sets used with SMP/E Release 5, or later, can be processed by SMP/E Release 8.1 and do not need to be converted.

### Format of MLPA Member (CBIPO Dialogs)

When creating an IEALPxx PARMLIB member, the CBIPO dialogs determine the correct format to use, rather than using the MVS/ESA SP Version 4 format as the default.

### Format of Tables in the CIDTABL Data Set (CBIPO Dialogs)

The CBIPO dialogs now require the tables in the CIDTABL data set to be in a new format. The tables for CBIPOs at spin level 94A or later are in the required format, but tables for pre-94A CBIPOs need to be migrated to the new format. To help users with existing CIDTABL data sets containing tables for pre-94A CBIPOs, the CIDCGCVT CLIST is provided in the SMP/E CLIST data set (GIM.SGIMCLS0) to migrate those tables to the new format.

### FunctionPac Support (CBIPO Dialogs)

The CBIPO dialogs can be used to install or reinstall FunctionPacs that are packaged as redistribution orders. The dialogs can also redistribute or connect an installed FunctionPac, provided that the table and skeleton information needed by the CBIPO dialogs has been defined.

### GENERATE Command

The SMP/E GENERATE command improves the system generation (SYSGEN) process by eliminating the need to separately reinstall products that do not have SYSGEN support. GENERATE can be used to do the following:

- Install all the products defined in a given target zone
- Install products provided on an MVS Custom-Built Installation Process Offering (CBIPO)
- Reinstall products not included when a generation procedure (such as SYSGEN or IMSGEN) replaced the system

The GENERATE command now provides a new operand, REPLACE, to create copy and link-edit jobs that replace existing members and load modules.

### GROUP Processing Improvements

During APPLY or ACCEPT processing, SMP/E can now more easily resolve requisites with the GROUPEXTEND operand. SMP/E generally resolves requisites by including the specified SYSMODs. Occasionally, a requisite is not available, but a SYSMOD that supersedes it is available. GROUPEXTEND tells SMP/E to search for a SYSMOD that supersedes the requisite and to include that SYSMOD during ACCEPT or APPLY processing.

---

## Hierarchical File Systems

SMP/E can now install and maintain elements and load modules residing in a *hierarchical file system (HFS)*. Here are some highlights of this support:

- DDDEF entries can be used to allocate paths within a hierarchical file system.
- A new type of modification control statement, ++HFS, is provided for packaging elements that reside in a hierarchical file system.

The PARM operand has been added to the ++HFS statement so that the hierarchical file system file access bits (permission bits) can be set at installation instead of as a postinstallation task. To take advantage of this support, the SMP/E user installing a SYSMOD containing this change must have the proper authority to set access bits through the mechanism employed by BPXCOPY.

- Processing of SMP/E commands has been changed to provide additional support. This includes changes to JCLIN processing of link-edit steps for load modules residing in a hierarchical file system.
- A new type of utility, the HFS copy utility, is recognized by SMP/E. The associated UTILITY entry is pointed to by the HFSCOPY subentry in the OPTIONS entry. \_

**Migration Note:** OPTIONS entries containing the new HFSCOPY subentry cannot be used by SMP/E Release 7 or earlier.

---

## High-Level Qualifier for RELFILE Data Sets

SMP/E now supports the use of a single, common, high-level qualifier for SYSMODs packaged in RELFILE format, instead of using the ID of the SYSMOD as the high-level qualifier. In the past, names of RELFILE data sets were in the format *sysmodid.Fn*. The names of RELFILE data sets using this new high-level qualifier are in the format *hlq.sysmodid.Fn*.

This support reduces the number of high-level qualifiers that must be defined to RACF (or another security-management program), as well as the need to extend security management definitions each time new products are to be installed.

**Note:** The naming convention used for SMPTLIB data sets allocated by SMP/E remains unchanged—those data set names do **not** include the new high-level qualifier used for RELFILE data sets.

---

## ISPF Version 2 Release 3 and ISPF/PDF Version 2 Release 3 Required

The SMP/E dialogs now require ISPF Version 2 Release 3 and ISPF/PDF Version 2 Release 3.

---

## Japanese Messages and Dialogs

In certain areas, SMP/E is now available in Japanese, as well as English. This support is provided by separate Japanese and English features for SMP/E.

- The Japanese feature provides SMP/E messages and dialogs in Japanese.
- The English feature provides SMP/E messages and dialogs in English.

Contact your local branch office for details on the availability of the Japanese feature.

---

### JCLIN Reports

SMP/E provides a summary report of the results of JCLIN processing. This includes changes, additions, and deletions to the CSI, as well as any EXEC statements that were ignored because the program or procedure specified was not recognized.

Besides this report, there is also a cross-reference report, which lists each macro, module, source, load module, and distribution library, and the pages in the summary report where each appears.

---

### JCLIN Saved at ACCEPT Time

When a SYSMOD containing inline JCLIN is installed in a distribution library, SMP/E can save the JCLIN data in the associated distribution zone. To do this, specify the ACCJCLIN indicator in the distribution zone before accepting the SYSMOD. The information provided by the JCLIN can be copied into the target zone and can be used by the GENERATE command to provide a SYSGEN-like installation process for products without SYSGEN support.

---

### Job Management Function (CBIPO Dialogs)

The CBIPO dialogs provide a job management function, which can be used to:

- Edit jobs before they are submitted.
- Monitor the status of jobs in a job stream.
- Exit the dialogs while jobs are running.
- Resume a job stream that has stopped partway through. (This formerly had to be done from outside the dialogs).

---

### Link-Edit Automatic Library Call Support

SMP/E now allows load modules to contain modules from multiple products without explicitly specifying those modules on INCLUDE statements in link-edit steps. This implicit inclusion of modules, which is done through SYSLIB DD statements in JCLIN link-edit steps, can be useful for products that:

- Are written in a high-level language and, as a result, include modules from libraries (such as compiler libraries) that are owned by a different product
- Make use of a callable-services interface provided by another product
- Need to include stub routines or interface modules from different products that may reside in other zones

To support such products, SMP/E can now use the CALL option and a specified SYSLIB allocation when invoking a link-edit utility. This allows the link-edit utility to include the necessary modules, reducing the need for postinstallation steps that must be run outside of SMP/E to accomplish the same task.



The LMOD entry can now contain a new subentry list (CALLLIBS) to record the SYSLIB allocation specified for a load module.

**Migration Note:** Target zones containing LMOD entries updated by SMP/E Release 8 cannot be used by SMP/E Release 7 or earlier, because the structure of the LMOD entries has changed to support the new CALLLIBS subentry list. LMOD entries are typically updated when JCLIN that defined the load module is processed.

When SMP/E services a load module built using the link-edit automatic library call feature, it rebuilds the load module without using the old load module as input. Instead, SMP/E uses as input a second load module that includes only the explicitly-defined modules from the original load module. This allows the automatic library call option to implicitly include modules at their latest service level.

SMP/E's link-edit support for the link-edit automatic library call feature is different from the SMP/E LINK command in two major respects:

- Defining the modules to be included

The automatic library call option is best used when a load module must include many modules from other products, and it is difficult and error-prone (and perhaps impossible) to define all the modules to be included.

The LINK command is more appropriate when a load module needs to include only a few specific modules from other products.

- Servicing load modules containing modules from other products

When a load module is built using the automatic library call option, SMP/E cannot completely service the load module because it does not know the full content of the load module. Specifically, the load module is not automatically rebuilt when an implicitly-included module is serviced. However, the new REPORT CALLLIBS command can be used to identify and relink such load modules. REPORT CALLLIBS is described in more detail below.

When a load module is updated with the LINK command, SMP/E knows the complete content of the load module and can automatically maintain the load module during subsequent APPLY and RESTORE processing.

---

## Messages Rewritten

The SMP/E messages have been rewritten to make them easier to use. In addition, the message identifiers (IDs) have been expanded from 6 to 8 characters, and message severity is now indicated by a letter instead of a number. Most messages still start with the same ID they used to have. In a few cases, a single message is now used for duplicate messages—two or more messages with different message IDs but essentially similar text and purpose.

Some messages are longer than in earlier releases of SMP/E. Therefore, you may need to increase the size of any data sets used for SMP/E messages (such as SMPDOUT, SMPLOG, and SMPLOGA). How much more space you need depends on the current size of these data sets and which messages are issued. To start, however, you may want to allocate new data sets that are twice the size of the old ones.

---

## MOVE/RENAME/DELETE Processing

A SYSMOD can contain control statements that move, delete, or rename system elements. When it installs such a SYSMOD, SMP/E provides a summary report of any changes made to system elements as a result of those control statements.

---

## MVSCP Elimination (CBIPO Dialogs)

For CBIPO orders containing MVS/ESA SP Version 5 and above, the MVSCP options of the CBIPO dialogs are not allowed; only the IODF path is supported.

If a customer with an MVSCP deck (or a combined MVSCP/IOCP deck) is installing an order that uses IODF data sets for its I/O definitions and the order is CBIPO spin level 94B minimum or later, the CBIPO dialogs prompt the user for information about the existing I/O definitions so that a batch job can be created to migrate the I/O configuration information to an IODF data set.

---

## MVS/370 Not Supported After SMP/E Release 8.1

SMP/E Release 8.1 is the last SMP/E release to run on MVS/370.

---

## New Packaging

Data elements are now used to package SMP/E. As a result, you cannot use SMP4 or a release of SMP/E before Release 5 to install SMP/E.

---

## Obsolete MCS Types and Operands Deleted

The following obsolete MCS types and operands are no longer supported:

++UPDTE	Use ++MACUPD instead.
ASMLIB	This operand was specified on ++MAC and ++MACUPD. Use DISTSRC instead.
BASE	This operand was specified on ++MAC, ++MACUPD, ++SRC, and ++SRCUPD. It is not needed under SMP/E.
DISTOBJ	This operand was specified on ++MAC, ++MACUPD, ++SRC, and ++SRCUPD. Use DISTMOD instead.

---

## Panel Identifier for Dialogs

The primary panel of the SMP/E dialogs displays the level of SMP/E that is currently installed.

---

## Performance Improvement: Automatically Using LSR

When running on MVS/ESA systems, SMP/E now automatically takes advantage of the local shared resource (LSR) feature of VSAM. This reduces the number of times SMP/E must access data when it is reading CSI data sets. As a result, SMP/E performance is improved for commands such as APPLY, APPLY CHECK, ACCEPT, ACCEPT CHECK, and especially LIST.

### Notes:

1. This performance improvement also occurs automatically on MVS/XA systems if the Data Facility Product (DFP) is at or beyond the minimum required service level (DFP 2.3 with PTFs UY22823, UY27311, UY28559, UY28999, and UY35924).
2. The improvement does not occur on MVS/370 systems at all, or on MVS/XA systems where DFP is not at or beyond the minimum required service level.
3. You may not experience any noticeable improvement in performance if you are implementing other buffering methods or batch LSR through JCL, as described in APAR OY24097. Your implementation of batch LSR overrides SMP/E's use of LSR. However, because SMP/E Release 8.1 makes use of this function through normal processing, you can remove the JCL references once SMP/E Release 8.1 is in production.

---

## Performance Improvements

SMP/E has improved its CPU utilization for processing APPLY and ACCEPT commands. Performance for CHECK processing has also been improved.

An appendix has been added to the *SMP/E R8.1 Reference* manual to describe SMP/E's implementation of local shared resources (LSR), as well as how the batch local shared resources (BLSR) subsystem can be used with expanded storage hiperspaces (instead of SMP/E's implementation of LSR) to improve SMP/E performance during APPLY and ACCEPT processing for a large number of changes.

---

## RACF Protection of Dynamically Allocated Data Sets

When a new data set is dynamically allocated, it can be RACF-protected if the PROTECT indicator is set in its DDDEF entry.

---

## RACF to Permit Read-Only Access to CSI Data Sets

RACF can now be used to permit read-only access to CSI data sets. This allows users to run commands that only read the CSI (such as LIST or REPORT), while preventing those users from updating the CSI.

### Regressed Subentries Deleted

The SYSMOD entry no longer contains subentries to indicate which elements supplied by this SYSMOD have been regressed by other SYSMODs.

### Receiving RELFILE Data Sets from DASD

In the past, the SMP/E RECEIVE command could process RELFILE data sets only if they were on tape. Now, SMP/E can process RELFILE data sets located on DASD as well.

### Recovery for Data Sets That Run Out of Space

Recovery processing for data sets that have run out of space has been improved in the following ways:

- Extended retry processing

When several members of a given data set need to be updated, SMP/E often “batches” those updates in a single utility call. When the utility attempts to update those members, the data set may run out of space (experience an x37 abend). In that case, if the user has requested retry processing, SMP/E compresses the data set and retries the utility.

In the past, if compressing a data set that had run out of space did not reclaim enough space, SMP/E terminated the job. Now, if compression fails to remedy the problem, SMP/E “debatches” the updates for that data set and retries the utility separately for each member in the data set, compressing the data set again if necessary. If this final attempt fails, the resulting x37 abend is treated as an unacceptable utility return code, and processing continues for other eligible updates.

- Excluding specific data sets from retry processing

It is now simpler for you to request retry processing when you want all but a few specific data sets to be included. In the past, the only way to do this was to specify each desired data set as a RETRYDDN value in the OPTIONS entry. Now you can achieve the same result by using the new EXRTYDD subentry for the OPTIONS entry along with RETRYDDN(ALL). Follow these steps:

1. Specify RETRYDDN(ALL) to include all your data sets in retry processing.
2. Specify EXRTYDD and the ddnames of the specific data sets to be excluded from retry processing.

**Migration Note:** OPTIONS entries containing the new EXRTYDD subentry cannot be used by SMP/E Release 7 or earlier.

### REJECT Improvements

SMP/E can now reject SYSMODs that satisfy very specific conditions. For example, it can reject SYSMODs for a specific function or reject SYSMODs for a function that is not part of the system. Also, more than one distribution zone or target zone can be checked before a SYSMOD is rejected.

HOLDDATA can be deleted even if the associated SYSMOD entry no longer exists. To do this, specify the HOLDDATA operand.

The new REJECT Summary report lists the SYSMODs that have been processed and explains why SYSMODs that were candidates for rejection were not rejected. In addition, statistics are included with the REJECT completion message to summarize how many SYSMOD entries, HOLDDATA entries, and FMIDs have been rejected.

---

## REPORT CALLLIBS Command

The REPORT command can now be used to identify and relink load modules when implicitly-included modules in a particular library are updated. Specifically, the REPORT CALLLIBS command tells you which load modules have a CALLLIBS subentry list in their LMOD entry (and therefore use the automatic library call option to implicitly include modules from a specified library). If requested, REPORT CALLLIBS also creates jobs to relink the load modules that are reported on. A separate job is created for each affected zone.

This command is helpful after a particular compiler library or callable-service data set has been upgraded with service or a new release. By specifying the upgraded library on the REPORT CALLLIBS command, you can find out which load modules for other products implicitly include modules from the upgraded library. By running the jobs created by REPORT CALLLIBS, you can relink these load modules so they include the latest routines from the upgraded library.

---

## Root Cause Analysis

To simplify the process of determining the errors that caused SYSMODs to fail, SMP/E provides a new report, the Causer SYSMOD Summary report. This report summarizes the messages describing the actual errors that caused the SYSMODs to fail, and that must be fixed in order to resolve the problem. The report filters out subsequent messages generated by these initial errors.

The SYSMOD Status report has also been improved:

- For each SYSMOD that failed, the report indicates the ID of the SYSMOD that caused the failure.
- A new statistics section summarizes the number of SYSMODs successfully processed and the number of SYSMODs that were not.

In addition, messages have been added, reworded, and deleted to provide better information for determining the cause of SYSMOD failure.

---

## SMPLOG Improvements

The SMPLOG is now a variable blocked data set. By specifying the block size, you can both reduce the time SMP/E spends writing data to the log and make more efficient use of DASD space.

In addition, you can define a second log data set to hold log information that does not fit in the primary log data set. This prevents the machine-readable data from being lost when the primary data set is full.

---

### SMPTLIB Improvements

SMP/E now automatically catalogs and uncatalogs any SMPTLIB data sets it allocates to receive SYSMODs that are packaged in RELFILE format. With this support, you no longer have to preallocate and catalog the data sets before receiving SYSMODs. SMP/E catalogs the SMPTLIB data sets when it allocates them and uncatalogs them when they are deleted.

When the SMPTLIB data sets are allocated by SMP/E during RECEIVE processing, dynamic allocation is used so these data sets can be RACF-protected. In addition, DDDEF entries can now be used to dynamically allocate the SMPTLIBs.

---

### SMPWRK5 Deleted

The SMPWRK5 data set is no longer used by SMP/E. SMPWRK5 had been used during APPLY processing when the modules to be link-edited into a given load module were not all in the same SMPTLIB data set. SMP/E now allocates its own data sets for this processing and no longer requires an SMPWRK5 data set.

---

### SMPWRK6 Data Set Required

The SMPWRK6 data set is required to apply or accept data elements that are packaged inline.

---

### Source ID Changes

- A SYSMOD can have more than one source ID.
- When SYSMOD entries are converted from SMP4 to SMP/E, a source ID of SMPECNV is automatically assigned to them.
- On the ACCEPT, APPLY, LIST, REJECT, and UNLOAD commands, you can now use a “wildcard” format to implicitly specify source IDs. For example, you can use *PUT\** to specify all source IDs beginning with the characters *PUT*.

---

### Source ID Exclusion

On the ACCEPT, APPLY, LIST, and UNLOAD commands, you can now use the new EXSRCID operand to indicate that all SYSMODs assigned a given source ID should be excluded from processing. This way, you do not have to exclude them individually.

---

## Source ID Report

The REPORT command can now be used to determine which source IDs are on your system. Specifically, the REPORT SOURCEID command tells you which source IDs are assigned to the SYSMODs in a given zone.

---

## Suppression of Member Names for Copy Processing

Through a new subentry in the UTILITY entry, SMP/E allows you to suppress the listing of member names during all SMP/E processing that invokes the copy or compress utility. With this new subentry, you can control the generation of the LIST=NO operand specification on the COPY control statement and eliminate unwanted output from invocations of the copy or compress utility.

**Migration Note:** UTILITY entries containing the new LIST subentry cannot be used by SMP/E Release 7 or earlier.

---

## Symbolic Substitution in Data Set Names (CBIPO Dialogs)

The Tailor and Build options of the CBIPO dialogs now support the specification and resolution of data set names that use the &SYSNAME symbol. PARMLIB support has been changed to allow the &SYSNAME symbol in data set names for LOGREC, SMF data sets, STGINDX, and PAGE data sets. This support for the &SYSNAME symbol in data set names facilitates the sharing of a master catalog in a parallel sysplex environment, thus easing the management of multiple images in that environment.

**Migration Note:** An MVS/ESA SP Version 5 redistribution order that has been tailored using symbols cannot be installed by the CBIPO dialogs in SMP/E Release 8 or earlier. This redistribution order is installable only on a system running SMP/E Release 8.1.

---

## SYSGEN No Longer Supported for SMP/E

SMP/E Release 8.1 cannot be installed as part of the MVS system generation (SYSGEN) process. The SMP/E SYSGEN macro (SGGIMSMP) should no longer be included in SYSGEN decks or used to generate SMP/E outside of the SYSGEN process. If you are doing a SYSGEN to build a system, you can use output from the SMP/E GENERATE command to install SMP/E in the target libraries.

---

## SYSMOD Selection

SMP/E provides greater flexibility in specifying the SYSMODs to be processed.

- You can process only SYSMODs that are applicable to specified functions. To do this, specify the desired functions on the FORFMID operand of the command being processed.
- You can process all SYSMODs of a specific type, without having to identify each individual SYSMOD. To do this, specify the desired SYSMOD types on the command being processed.
- You can ensure that requisites needed by the SYSMODs being installed are also included. To do this, specify the GROUP or GROUPEXTEND operand on

the command being processed. GROUP includes SYSMODs that are specifically defined as requisites. Occasionally the requisite is not available, but a SYSMOD that supersedes the requisite is available. GROUPEXTEND tells SMP/E to search for a SYSMOD that supersedes the requisite and to include this SYSMOD during ACCEPT or APPLY processing.

- You can process all SYSMODs associated with a given source. To do this, specify the identifiers for the desired sources on the command being processed. These source IDs can be associated with SYSMODs by the RECEIVE command, the ++ASSIGN statement, or CONVERT processing. A SYSMOD can have more than one source ID.

Because of these selection options and SMP/E's exception SYSMOD processing, you no longer need to perform extensive research before receiving or installing SYSMODs. The SYSMODs can be stored in the SMPPTS data set and can be selectively processed by subsequent commands. This provides a convenient way of subsetting SMP/E processing without the labor-intensive research and job stream tailoring required with SMP4 to generate and maintain extensive EXCLUDE and SELECT lists. In addition, mass mode SMP/E operations can be conveniently reduced to more manageable units.

SMP/E RECEIVE processing is also changed so that, unless specifically requested, no modification control statements are listed as part of the RECEIVE operation. Instead, you can use the LIST command to obtain them from the SMPPTS. This is helpful to get a copy of the "cover letter" for a PTF.

For each SYSMOD SMP/E successfully receives, one message is printed indicating a successful RECEIVE. No messages are produced for SYSMODs not applicable to the system. This reduces the excessive output produced by SMP4, which included partial listings of SYSMODs not applicable to the system.

For a SYSMOD found with syntax errors, SMP/E prints an error message, and the SYSMOD is not received.

At the completion of RECEIVE processing, SMP/E produces a RECEIVE Summary report containing information for each SYSMOD in the input file.

---

## SYSMOD Status Report Accuracy

The accuracy of the SYSMOD Status report has been improved for the listing of unsatisfied requisite SYSMOD and hold conditions. SMP/E has been enhanced to search beyond the initial point of SYSMOD termination and report on additional requisite SYSMOD and hold conditions that are unresolved for the failing SYSMOD. These conditions, identified by a dash (-) in the SYSMOD Status report, are also identified by additional GIM359xx messages. This can improve a system programmer's productivity by reducing the number of times it is necessary to run a mass APPLY CHECK or ACCEPT CHECK in order to resolve all conditions required to install the desired SYSMODs.



---

## UPGRADE Operand Deleted

The UPGRADE operand of the ACCEPT and APPLY commands is no longer supported. UPGRADE had been used along with the SELECT operand to reinstall function SYSMODs. Functions can still be reinstalled with SELECT and the REDO operand, which supports any type of SYSMOD.

---

## Utility Error Isolation

SMP/E now writes a time, date, and sequence number stamp on completion messages for copy, link-edit, superzap, and update processing. Because the same sequence number is included in the utility SYSPRINT output, it can be used as an index to find the desired information in the SYSPRINT data set. The completion message for assembler processing includes a time and date stamp, which is included in the assembler output and can aid in finding that output.

In addition, there are changes in how the copy utility and link-edit utility are called. For calls to the copy utility, only members with the same input and output data sets are processed during a single call. If a batched call to the link-edit utility fails, the call is unbatched to isolate where the error occurred.

---

## Virtual Storage Constraint Relief

SMP/E can now run on an MVS/XA system using less than 4 megabytes of storage below the 16-megabyte line.

---

## VSAM Data Sets

SMP/E control information is maintained in VSAM data sets, rather than the partitioned data sets used by SMP4. The VSAM data sets are known as the *consolidated software inventory (CSI)*. These inventory data sets contain information on system structure, system function and service levels, and other control information required for SMP/E processing. Within any one VSAM data set, multiple products, subsystems, and systems can be represented as separate zones, which are analogous to the SMP4 CDS and ACDS data sets. Both system target library and distribution library information can be maintained within one inventory data set as separate zones. In addition to these target and distribution zones, each CSI contains a global zone, which defines the other zones in the CSI and describes SYSMODs waiting to be installed, system utilities to be invoked during SMP/E processing, and DD definitions for use by dynamic allocation.

The SMP4 data sets that must be converted to the VSAM CSI structure are:

- Control information from the PTF temporary store (PTS) data set. The actual text of SYSMODs continues to be maintained in a PTS partitioned data set in SMP/E.
- The control data set (CDS).
- The alternative control data set (ACDS).
- The conditional requisite queue (CRQ).
- The alternate conditional requisite queue (ACRQ).

The other SMP data sets require no conversion. SMP/E includes data set conversion and data set administration facilities.

The VSAM CSI implementation provides:

- An architectural base for new SMP/E functions
- Increased data set management capability
- Simplification and consolidation of SMP/E data
- Inquiry capability against the SMP/E data

---

## VS1 No Longer Supported

SMP/E Release 3, and later, does not run on VS1. (These releases can still be used to install and maintain products on a VS1 system.)

---

## Zone Descriptions

SMP/E allows you to include descriptive text in a global zone, a distribution zone, or a target zone. This description is solely for your own use; SMP/E does not refer to it.

---

## Zone Editing

The SMP/E ZONEEDIT command helps you quickly and easily change subentries in all the DDDEF or UTILITY entries contained in a zone. With ZONEEDIT you can make global changes with one command instead of having to individually update each entry with the same change.

---

## Zone Utility Commands

The following SMP/E commands help you manage the SMP/E CSI data sets at the zone level:

- ZONECOPY copies an entire zone from one CSI to another CSI, such as when copying a distribution zone to a target zone in preparation for a SYSGEN.
- ZONEDELETE deletes a specified target zone or distribution zone from the CSI in which it was contained.
- ZONEEXPORT copies a zone from a CSI to a sequential data set, thus creating a transportable, backup copy of a zone.
- ZONEIMPORT loads a previously exported zone into a CSI.
- ZONEMERGE copies an existing zone into an empty zone, or merges two existing zones.
- ZONERENAME renames an existing target zone or distribution zone, or changes a distribution zone to a target zone.

---

## **ZONESETs with Mixed Zones**

A ZONESET can now contain both target and distribution zones, rather than only zones of a single type.

---

## **4-Digit Device Addresses (CBIPO Dialogs)**

The Build option of the CBIPO dialogs allows 4-digit device addresses to be entered wherever device addresses can be specified. If the address has not been genned, it is rejected.

If the order being processed contains MVS/ESA SP Version 5, the Tailor option of the CBIPO dialogs allows 4-digit device addresses to be entered on the stand-alone dump panels. If the order contains a lower level of MVS, 4-digit device addresses are not allowed.



## Appendix H. Compatibility with Previous SMP/E Releases

For compatibility, all SYSMOD input (modification control statements) acceptable to SMP Release 4 (SMP4) and previous releases of SMP/E is acceptable to SMP/E Release 8.1.

New releases of SMP/E generally add to or enhance the information stored in SMP/E data sets. This new or enhanced data is not always compatible with previous releases of SMP/E. Results are unpredictable when a prior release of SMP/E encounters data introduced by SMP/E Release 8.1 function. Table 44 summarizes restrictions on processing SMP/E entries and other data with earlier releases of SMP/E.

**Note:** SMP/E issues message GIM44330 when it finds records in the CSI that it does not recognize. This occurs when an earlier release of SMP/E is used to process a CSI that has been updated by a later release. Table 44 describes these situations, including possible circumventions.

Table 44 (Page 1 of 2). Processing SMP/E Data with Earlier Releases of SMP/E

Data	Restriction	Circumvention	References
<b>Restrictions Introduced in SMP/E Release 8.1</b>			
CBIPO dialogs: redistribution orders	An MVS/ESA SP Version 5 redistribution order that has been tailored so that data set names contain symbols cannot be installed by the CBIPO dialogs in SMP/E Release 8 or earlier.	None. This redistribution order is installable only on a system running SMP/E Release 8.1.	<i>SMP/E R8.1 CBIPO Dialogs User's Guide</i>
<b>Restrictions Introduced in SMP/E Release 8</b>			
CBIPO dialogs: CIDTABL data set	To use an existing CIDTABL data set that contains data for pre-94A CBIPOs, you must use the CIDCGCVT CLIST provided with SMP/E Release 8.1 to migrate the tables in the data set to the new format required by the dialogs.	None.	<i>SMP/E R8.1 CBIPO Dialogs User's Guide</i>
LMOD entries	Target zones containing LMOD entries updated by SMP/E Release 8 or later cannot be used by SMP/E Release 7 or earlier, because the structure of the LMOD entries has changed to support the CALLLIBS subentry list. LMOD entries are typically updated when JCLIN that defined the load module is processed.	None. You must use SMP/E Release 8 or later to process the affected zones.	<i>SMP/E R8.1 Reference</i>

## Compatibility with Previous Releases

Table 44 (Page 2 of 2). Processing SMP/E Data with Earlier Releases of SMP/E

Data	Restriction	Circumvention	References
OPTIONS entries: EXRTYDDN	OPTIONS entries that contain the new EXRTYDD subentry to exclude data sets from retry processing cannot be used by SMP/E Release 7 or earlier.	Do one of the following: <ul style="list-style-type: none"> <li>• Use a different OPTIONS entry.</li> <li>• Using SMP/E Release 8 or later, remove the EXRTYDD subentry from the OPTIONS entry with the SMP/E Administration dialogs or the UCLIN command.</li> </ul>	<i>SMP/E R8.1 Reference</i> <i>SMP/E R8.1 User's Guide</i>
OPTIONS entries: HFSCOPY	OPTIONS entries that contain the new HFSCOPY subentry to identify a UTILITY entry for the hierarchical file system copy utility cannot be used by SMP/E Release 7 or earlier.	Do one of the following: <ul style="list-style-type: none"> <li>• Use a different OPTIONS entry.</li> <li>• Using SMP/E Release 8 or later, remove the HFSCOPY subentry from the OPTIONS entry with the SMP/E Administration dialogs or the UCLIN command.</li> </ul>	<i>SMP/E R8.1 Reference</i>
UTILITY entries: LIST	UTILITY entries that contain the new LIST subentry to suppress the listing of member names cannot be used by SMP/E Release 7 or earlier.	Do one of the following: <ul style="list-style-type: none"> <li>• Use a different UTILITY entry.</li> <li>• Using SMP/E Release 8 or later, remove the LIST subentry from the UTILITY entry with the SMP/E Administration dialogs or the UCLIN command.</li> </ul>	<i>SMP/E R8.1 Reference</i>
<b>Restrictions Introduced in SMP/E Release 7</b>			
LINK command	Zones updated by the LINK command (introduced in SMP/E Release 7) cannot be used by SMP/E Release 6 or earlier.	None. You must use SMP/E Release 7 or higher to process the affected zones.	<i>SMP/E R8.1 Reference</i>

---

## Appendix I. Performance Improvements through Local Shared Resources (LSR)

This appendix briefly describes how SMP/E performance is improved through the use of local shared resources (LSR). For more information on LSR, see the *MVS/ESA Batch Local Shared Resources Subsystem* manual, GC28-1059 (MVS/SP Version 3), or the *MVS/ESA Application Development Guide: Batch Local Shared Resources* manual, GC28-1672 (MVS/ESA SP Version 4).

---

### SMP/E's Automatic Use of Local Shared Resources (LSR)

When running on MVS/ESA systems, SMP/E automatically takes advantage of the local shared resource (LSR) feature of VSAM. This reduces the amount of I/O needed in order to read CSI data sets to satisfy SMP/E's access to CSI information. As a result, SMP/E performance is improved for commands such as APPLY, APPLY CHECK, ACCEPT, ACCEPT CHECK, and especially LIST.

This performance improvement occurs automatically on MVS/XA systems if the Data Facility Product (DFP) is at or beyond the minimum required service level (DFP 2.3 with PTFs UY22823, UY27311, UY28559, UY28999, and UY35924). The improvement does not occur on MVS/370 systems at all, or on MVS/XA systems where DFP is not at or beyond the minimum required service level.

You may not experience any noticeable improvement in performance if you are currently implementing other buffering methods or batch LSR through JCL, as described in APAR OY24097. Your implementation of batch LSR overrides SMP/E's use of LSR. If you are using batch LSR, you may want to remove the JCL references once SMP/E Release 8.1 is in production, in order to see if the SMP/E implementation of LSR provides equivalent or improved performance.

#### Notes:

1. When opening a CSI data set for read access, SMP/E uses BUFNI=100 to get index buffers and BUFND=25 to get data buffers. By implementing batch LSR through JCL, as described in APAR OY24097, you can override the values specified by SMP/E.
2. When you override SMP/E's implementation of LSR by using batch LSR through JCL, exercise caution if using the SHRPOOL operand. Use of the SHRPOOL operand may prevent SMP/E from opening a CSI when multiple CSI data sets with different CISIZE values are being processed.

---

### Improving APPLY and ACCEPT Processing for a Large Number of Changes

If you have a large number of changes for SMP/E APPLY or ACCEPT processing, you can reduce the elapsed time for processing by using the batch local shared resources (LSR) subsystem together with expanded storage hiperspaces (instead of SMP/E's implementation of LSR).

---

To do this, you use JCL statements instead of dynamic allocation to define the CSI data sets containing the zones required for processing. For each CSI data set, you need to provide two DD statements:

- The first DD statement is used by SMP/E to enqueue on the data set (to protect it from simultaneous updates) and to trigger batch LSR. This DD statement uses the DDNAME parameter to point to the second DD statement.
- The second DD statement is used by batch LSR to open the CSI data set.

Both DD statements must specify the same DSN value. The following example shows how this can be done:

```
//SMPCSI DD DSN=dataset1.CSI,DISP=SHR,  
//      SUBSYS=(BLSR,'DDNAME=MYSMPCSI',  
//            'HBUFND=value','HBUFNI=value')  
//MYSMPCSI DD DSN=dataset1.CSI,DISP=SHR  
//*  
//tgtzone DD DSN=dataset2.CSI,DISP=SHR,  
//      SUBSYS=(BLSR,'DDNAME=MYTG1',  
//            'HBUFND=value','HBUFNI=value')  
//MYTG1 DD DSN=dataset2.CSI,DISP=SHR
```

**Notes:**

1. If the target zone and global zone exist in the same CSI, *dataset1* and *dataset2* refer to the same data set.
2. HBUFND and HBUFNI are used instead of BUFND and BUFNI to indicate the use of hiperspace in expanded storage (if available).
3. *tgtzone* is the name of the target zone specified on the SET BDY command to be processed.
4. The DSN parameter must be specified on all DD statements to ensure internal enqueue protection.
5. This JCL can be used with SMP/E Release 5, or later.



---

## Appendix J. GIMGNIAP: HFS Copy Utility Invocation Program

The SMP/E GENERATE command creates a job stream that builds a set of target libraries from a set of distribution libraries. When elements need to be installed in a hierarchical file system (HFS), the GENERATE command builds an HFSINST job to invoke the HFS copy utility for those elements. Each job step in the HFSINST job installs multiple HFS elements from multiple distribution libraries into a single target library. This is done by invoking the SMP/E program GIMGNIAP, which calls the HFS copy utility to do the actual installation.

Although GIMGNIAP is to be used only by SMP/E, you may need to understand the control statements passed to the program. For example, you may need to diagnose errors detected by GIMGNIAP (such as syntax errors or missing information). To help you with this task, this chapter describes:

- The control statements used to invoke GIMGNIAP
- The return codes issued by GIMGNIAP
- The JCL statements used in the HFSINST job that invokes GIMGNIAP

---

### Control Statements Used to Invoke GIMGNIAP

The GIMGNIAP program runs as a background job, and is driven using control statements that identify the following:

- The HFS copy utility to be invoked
- The distribution and target libraries to be used for each invocation of GIMGNIAP
- The HFS elements to be installed
- The parameters to be passed to the HFS copy utility

These control statements are created by SMP/E during GENERATE processing and are for use during GIMGNIAP processing. Each invocation of GIMGNIAP may install many HFS elements through multiple invocations of the HFS copy utility. Each invocation of the HFS copy utility by GIMGNIAP installs a single HFS element. These are the control statements used for input to a GIMGNIAP step:

- INVOKE
- COPY
- SELECT

### The INVOKE Control Statement

The INVOKE control statement identifies the HFS copy utility that GIMGNIAP should invoke. It specifies the following:

- The name of the HFS copy utility
- The highest acceptable return code from the HFS copy utility.
- The ddname for HFS copy utility print output

The INVOKE control statement is always produced. There is one INVOKE control statement for each job step (that is, one INVOKE control statement for each invo-

cation of GIMGNIAP). The INVOKE control statement must be the first control statement and must be a single-card image (one 80-byte record). The following operands are required on the INVOKE statement:

**HFSCOPY**

is the name of the HFS copy utility load module that GIMGNIAP is to invoke. The HFSCOPY value is either the value of the NAME operand in the UTILITY entry in effect at GENERATE time or the default of BPXCOPY.

**RC**

is the highest acceptable return code from the HFS copy utility. The RC value is either the value of the RC operand in the UTILITY entry in effect at GENERATE time or the default of 0.

**PRINT**

is the ddname that is to be used for print output generated by the HFS copy utility. The PRINT value is either the value of the PRINT operand in the UTILITY entry in effect at GENERATE time or the default of SYSPRINT.

**Note:** If SYSTSPRT is specified as the PRINT value for the HFS copy utility, it is ignored and the default of SYSPRINT is used instead.

Table 45 shows HFS copy utility UTILITY entries with varying pieces of information and the corresponding INVOKE statement generated for each case.

<i>Table 45. Examples of INVOKE Statements Built from HFS Copy Utility UTILITY Entries</i>	
<b>UTILITY Entry Subentries</b>	<b>INVOKE Statement</b>
NAME(MYHFSCPY) RC(4) PRINT(MYPRINT)	INVOKE HFSCOPY(MYHFSCPY) RC(4) PRINT(MYPRINT)
RC(8)	INVOKE HFSCOPY(BPXCOPY) RC(8) PRINT(SYSPRINT)
PRINT(ABCOOPS)	INVOKE HFSCOPY(BPXCOPY) RC(0) PRINT(ABCOOPS)

**COPY Control Statement**

A COPY control statement identifies the input and output libraries to be used for the SELECT control statements that follow it. Each COPY statement identifies a distribution library that has HFS elements to be copied to a single target library. Multiple COPY statements can identify the same target library within the hierarchical file system.

A COPY control statement must follow an INVOKE or SELECT control statement and must be a single-card image (one 80-byte record). In addition, each COPY control statement must be followed by at least one SELECT control statement. The following operands are required on the COPY statement:

**FROMLIB**

is the ddname used by the HFS copy utility as its input file for the source of the HFS element (that is, the distribution library). The name can be 1 to 8 characters and must be composed of uppercase alphabetic, numeric, or national (\$, #, @) characters.

**TOLIB**

is the ddname used by the HFS copy utility as its output file identifying where the HFS element is to be installed (that is, the target library within the hierar-

chical file system). The name can be 1 to 8 characters and must be composed of uppercase alphabetic, numeric, or national (\$, #, @) characters.

## SELECT Control Statement

The SELECT control statement identifies the HFS element to be installed and the parameter list to be passed to the HFS copy utility to enable its installation. The SELECT control statement must follow a COPY control statement and can span multiple 80-byte records. The following operands are required on the SELECT statement:

### HFS

is the name of the HFS element to be installed into the hierarchical file system. The HFS operand must be the first operand on the SELECT statement.

### EPARM (LLOption1,option2,option3,...)

is the parameter list to be passed for this invocation of the HFS copy utility. The EPARM operand must be the last operand on the SELECT statement. The parameter list consists of an LL value and the actual execution parameters.

- LL represents a halfword hexadecimal length of the character string (the HFS copy utility execution parameters) immediately following it as part of the EPARM value. The length of the character string described does not include the opening parenthesis preceding the LL value nor the closing parenthesis following the last option specified.

Because the LL value is nondisplayable (ready-to-use) hexadecimal, it may appear as blanks or odd characters. This is valid data and should not be removed or modified.

- The *option* values are the execution parameters to be used by the HFS copy utility for this invocation of the utility. SMP/E always supplies execution parameters to the HFS copy utility, and the parameters are separated by commas with no intervening blanks.

If the HFSCOPY UTILITY entry that is in effect supplies execution parameters, these values precede the SMP/E-generated information. For example, suppose the UTILITY entry has the following values:

```
NAME MYHFSCPY
PRINT MYPRINT
PARM A-PARM-FOR-MYHFSCPY
```

The character string for the execution parameters would be generated as:

```
LLA-PARM-FOR-MYHFSCPY,ELEMENT(hfse1m1),TYPE(TEXT),LINK('linknm01'))
```

If a PARM value of *user\_info* is specified in the ++HFS MCS, the character string for the execution parameters would be generated as:

```
LLA-PARM-FOR-MYHFSCPY,user_info,ELEMENT(hfse1m1),TYPE(TEXT),LINK('linknm01'))
```

These are the parameters that are generated by SMP/E, using information in the HFS element entry:

### ELEMENT(element\_name)

the name of the HFS element to be installed in the hierarchical file system. The *element\_name* is an unquoted character string 1 to 8

bytes long. It is composed of uppercase alphabetic, numeric, or national (\$, #, @) characters.

**LINK('linkname1','linkname2','linkname3'...)**

the alternate names by which this element can be known in the target library within the hierarchical file system. A *linkname* can be up to 64 characters long and can contain special characters other than just uppercase alphabetic, numeric, or national (\$, #, and @) characters.

SMP/E always puts apostrophes around each *linkname* and separates multiple values with commas with no intervening blanks.

**TYPE(TEXT|BINARY)**

the installation format for the HFS element in the hierarchical file system. SMP/E generates this parameter from information stored in the HFS element entry in the target zone. If no setting exists for TEXT mode or BINARY mode, SMP/E does not pass this parameter to the HFS copy utility.

## Rules for GIMGNIAP Control Statements

The following rules apply to the format of INVOKE, COPY, and SELECT control statements. This information can help you diagnose errors detected by GIMGNIAP (such as syntax errors or missing information).

- **All control statements**

- Only the SELECT control statement can span multiple 80-byte records. The data for the INVOKE and COPY statements must be on one 80-byte record.
- Control statement verbs must appear in column 1. The allowable verbs are INVOKE, COPY and SELECT.
- The data for a control statement (either the initial 80-byte record or a continuation 80-byte record) must start in column 1 and end anywhere up to and including column 72.
- Data that is continued to another 80-byte record must extend through column 72 of the preceding 80-byte record.
- Control statement operands can appear anywhere and in any order on the 80-byte record. The only exception is the HFS operand, which must be the first operand on the SELECT statement.
- Columns 73 through 80 are ignored.
- No continuation character is used since the data is self-defining with regard to length.
- No comments are allowed.

- **SELECT control statements only**

- The HFS operand and the EPARM operand (up through the LL value) must appear on the first record with the SELECT verb.
- No blanks are allowed between the EPARM left parenthesis and the LL value.
- No blanks are allowed between the LL value and the execution parameters.

- The LL value must appear on the first record with the SELECT verb.
- No blanks are allowed between the execution parameters and the closing parenthesis for EPARM.

Figure 143 shows examples of invalid control statements:

Column 1	Column 72
v	v
<hr/>	
1. SELECT HFS(element) EPARM(LLELEMENT(element),TYPE(BINARY),LINK('linkname1','linkname2'))	
2. SELECT HFS(element) EPARM(LL ELEMENT(element),TYPE(BINARY),LINK('linkname1','linkname2'))	
3. SELECT HFS(element) EPARM(LLELEMENT(element),TYPE(BINARY) )	
4. SELECT HFS(element) EPARM(LLELEMENT(element),TYPE(BINARY),LINK('linkname1','linkname2'))	
5. SELECT HFS(element) EPARM(LLELEMENT(element),TYPE(BINARY),LINK('linkname1','linkname2'))	
6. SELECT HFS(element) EPARM(LLELEMENT(element),TYPE(BINARY),LINK('linkname1','linkname2'))	

Figure 143. Examples of Invalid Control Statements for GIMGNIAP

Figure 144 shows examples of valid control statements:

Column 1	Column 72
v	v
<hr/>	
1. SELECT HFS(element) EPARM(LLELEMENT(element),TYPE(BINARY),LINK('linkname1','linkname2'))	
2. SELECT HFS(element) EPARM(LL ELEMENT(element),TYPE(BINARY),LINK('linkname1','linkname2'))	

Figure 144. Examples of Valid Control Statements for GIMGNIAP

Figure 145 shows an example of the control statements for one invocation of GIMGNIAP:

Columns

```

-----1-----2-----3-----4-----5-----6-----7-----8
INVOKE HFSCOPY(BPXCOPY) RC(0) PRINT(SYSPRINT)
COPY FROMLIB(distlib1) TOLIB(tgthfs01)
SELECT HFS(hfse1m1) EPARM(LLELEMENT(hfse1m1),LINK('linkname01','linkname
that is longer than the first one and should be more user friendly'),TYPE(BINARY))
SELECT HFS(hfse1m2) EPARM(LLELEMENT(hfse1m2),LINK('linknm01'),TYPE(TEXT)
)
COPY FROMLIB(distlib2) TOLIB(tgthfs01)
SELECT HFS(hfse1m3) EPARM(LLELEMENT(hfse1m3),TYPE(TEXT))
SELECT HFS(hfse1m4) EPARM(LLELEMENT(hfse1m4),LINK('linknm01','linknm02',
'linknm03','linknm04','linknm05'))

```

Figure 145. Complete Example of Control Statements for Invoking GIMGNIAP

In Figure 145, the first COPY statement and the following two SELECT statements instruct GIMGNIAP to install (via two invocations of the HFS copy utility) elements *hfse1m1* and *hfse1m2* from distribution library *distlib1* into target library *tgthfs01* within the hierarchical file system.

The second COPY statement and the following two SELECT statements instruct GIMGNIAP to install (via two invocations of the HFS copy utility) elements *hfse1m3* and *hfse1m4* from distribution library *distlib2* into target library *tgthfs01* within the hierarchical file system. Note that *hfse1m3* has no linkname associated with it.

For an example of two job steps invoking GIMGNIAP and the input to the GIMGNIAP program for each, see “JCL Statements Used in the HFSINST Job” on page 875.

## Return Codes

To help you diagnose errors, GIMGNIAP issues messages and return codes. The messages are documented in the *SMP/E R8.1 Messages and Codes* manual. Here is a description of the return codes:

Return Code	Meaning
0	GIMGNIAP processing was successful. The message issued by GIMGNIAP states that the invocation of GIMGNIAP was successful.
8	A return code from the HFS copy utility was higher than the acceptable return code specified on the INVOKE control statement. At least one element was not installed correctly, although an attempt to install all HFS elements to this hierarchical file system target was made. You should review the HFS copy utility print output for the error encountered during the installation of the identified HFS element. After correcting the problem, you should rerun the job step.

Return Code	Meaning
12	GIMGNIAP encountered an invalid control statement. As a result, the HFS elements in that job step were not installed. Once a control statement error is encountered, no more HFS elements are processed for that job step. The statement must be corrected and the job step must be rerun. However, subsequent job steps will be processed. The message issued by GIMGNIAP indicates how many 80-byte records had been processed when the syntax error was encountered.
16	A severe error was encountered. As a result, the HFS elements in that job step were not installed. After correcting the problem, the job step must be rerun. However, subsequent job steps will be processed. The message issued by GIMGNIAP indicates the cause of the problem, such as an I/O error for the SYSIN data set, an open failure for the SYSIN data set, or absence of the HFS copy utility.

## JCL Statements Used in the HFSINST Job

JCL statements are created for the HFSINST job during the GENERATE process. Along with the GIMGNIAP control statements for each job step, the JCL is composed of:

- **A JOB statement.** The JOB statement describes the current installation-dependent parameters. The jobname is "HFSINST."
- **One or more EXEC statements.** The EXEC statement always specifies `PGM=GIMGNIAP`. The step name is the name of the target library that is within the hierarchical file system.
- **DD statements.** The DD statements define the data sets to be used by GIMGNIAP processing. The following ddnames are required:

<b>dlib</b>	is the file identified by a FROMLIB operand on a COPY control statement.
<b>syslib</b>	is the file identified by a TOLIB operand on a COPY control statement.
<b>print</b>	is the ddname used by the HFS copy utility for messages and processing information. If no DDDEF is found for the print file, the default of <code>SYSOUT=*</code> is generated. The ddname is the same as is specified for the PRINT operand of the INVOKE statement being used for the current invocation of GIMGNIAP.
<b>SYSIN</b>	is the input control stream for GIMGNIAP.

Figure 146 on page 876 shows an example of the JCL statements that could be generated to invoke GIMGNIAP processing.

```

-----1-----2-----3-----4-----5-----6-----7-----8
1 //HFSINST JOB 'accounting info',MSGLEVEL=(1,1)
2 //tghfs01 EXEC PGM=GIMGNIAP EXECIAP
//distlib1 DD distribution library DD info. distlib1
//distlib2 DD distribution library DD info. distlib2
//tghfs01 DD PATH='/Hierarchical/File/System/library1/DD/info/' tghfs01
//SYSPRINT DD SYSOUT=* DEFAULT
//SYSIN DD * DEFAULT
INVOKE HFSCOPY(BPXCOPY) RC(0) PRINT(SYSPRINT)
COPY FROMLIB(distlib1) TOLIB(tgthfs01)
SELECT HFS(hfse1m1) EPARM(LLELEMENT(hfse1m1),LINK('linkname01','linkname
that is longer than the first one and should be more user friendly'),TYPE(BINARY))
SELECT HFS(hfse1m2) EPARM(LLELEMENT(hfse1m2),LINK('linknm01'),TYPE(TEXT)
)
COPY FROMLIB(distlib2) TOLIB(tgthfs01)
SELECT HFS(hfse1m3) EPARM(LLELEMENT(hfse1m3),TYPE(TEXT))
SELECT HFS(hfse1m4) EPARM(LLELEMENT(hfse1m4),LINK('linknm01','linknm02',
'linknm03','linknm04','linknm05'))
/*
2 //tghfs02 EXEC PGM=GIMGNIAP EXECIAP
//distlib3 DD distribution library DD info. distlib3
//distlib4 DD distribution library DD info. distlib4
//tghfs02 DD PATH='/Hierarchical/File/System/library2/DD/info/' tghfs02
//SYSPRINT DD SYSOUT=* DEFAULT
//SYSIN DD * DEFAULT
INVOKE HFSCOPY(BPXCOPY) RC(0) PRINT(SYSPRINT)
COPY FROMLIB(distlib3) TOLIB(tgthfs02)
SELECT HFS(hfse1m3) EPARM(LLELEMENT(hfse1m3),LINK('linknm01','linknm02',
'linknm03','linknm04'),TYPE(TEXT))
SELECT HFS(hfse1m4) EPARM(LLELEMENT(hfse1m4),LINK('linknm01'),TYPE(TEXT)
)

```

Figure 146 (Part 1 of 2). Sample HFSINST Job for Invoking GIMGNIAP



```
COPY FROMLIB(distlib4) TOLIB(tgthfs02)
SELECT HFS(hfse1m5) EPARM(LLELEMENT(hfse1m5),TYPE(TEXT))
SELECT HFS(hfse1m6) EPARM(LLELEMENT(hfse1m6),LINK('linknm01','linknm02',
'linknm03','linknm04','linknm05'),TYPE(TEXT))
/*
```

**Notes:**

- 1** The **job name** “HFSINST” is generated during the job creation process. A single job installs all HFS elements.
- 2** The **job step name** on the EXEC statement is the target library within the hierarchical file system. Each job step installs multiple HFS elements from multiple distribution libraries into a single target library within the hierarchical file system.

Figure 146 (Part 2 of 2). Sample HFSINST Job for Invoking GIMGNIAP



---

## Appendix K. List of Macros Intended for Customer Use

Macro GIMMPUXP is provided as a Product-Sensitive programming interface for customers by SMP/E Release 8.1.

**Warning:** Do not use as programming interfaces any SMP/E Release 8.1 macros other than GIMMPUXP.



## Glossary

This glossary defines terms and abbreviations used in this publication. If you do not find the term you are looking for, refer to the index portion of this book, to prerequisite publications, or to the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

**Sequence of Entries:** For clarity and consistency of style, this glossary arranges the entries alphabetically on a letter-by-letter basis. In other words, only the letters of the alphabet are used to determine sequence; special characters and spaces between words are ignored.

**Organization of Entries:** Each entry consists of a single-word or multiple-word term or the abbreviation or acronym for a term, followed by a commentary. A commentary includes one or more items (definitions or references) and is organized as follows:

1. An item number, if the commentary contains two or more items.
2. A usage label, indicating the area of application of the term, for example, "In programming," or "In SMP/E." Absence of a usage label implies that the term is generally applicable to SMP/E, to IBM, or to data processing.
3. A descriptive phrase, stating the basic meaning of the term. The descriptive phrase is assumed to be preceded by "the term is defined as ..." The part of speech being defined is indicated by the opening words of the descriptive phrase: "To ..." indicates a verb, and "Pertaining to ..." indicates a modifier. Any other wording indicates a noun or noun phrase.
4. Annotative sentences, providing additional or explanatory information.
5. References, directing the reader to other entries or items in the dictionary.

**References:** The following cross-references are used in this glossary:

**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 glossary.

**Synonymous with.** This is a backward reference from a defined term to all other terms that have the same meaning.

**See.** This refers you 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.

**Deprecated term for or Deprecated abbreviation for.** This indicates that the term or abbreviation should not be used. It refers to a preferred term, which is defined in its proper place in the glossary.

**Selection of Terms:** A term is a word or group of words to be defined. In this glossary, the singular form of the noun and the infinitive form of the verb are the terms most often selected to be defined. If the term may be abbreviated, the abbreviation is given in parentheses immediately following the term. The abbreviation is also defined in its proper place in the glossary.

## A

**ACCEPT.** The SMP/E command used to install SYSMODs in the distribution libraries.

**accept.** In SMP/E, to install SYSMODs in the distribution libraries. This is done with the ACCEPT command.

**accepted SYSMOD.** A SYSMOD that has been successfully installed by the SMP/E ACCEPT command. Accepted SYSMODs do not have the ERROR flag set and are found as SYSMOD entries in the distribution zone.

**Access method services (AMS).** The system utility program used to support VSAM data sets.

**ACDS.** Alternative control data set.

**alternative control data set (ACDS).** An SMP4 data set used to record information about a distribution library.

**ACRQ.** Alternative conditional requisite queue.

**alternative conditional requisite queue (ACRQ).** An SMP4 data set used to record information about a distribution library.

**AMS.** Access method services.

**APAR.** Authorized program analysis report.

**APAR fix.** A temporary correction of a defect in an IBM system control program or licensed program that affects a specific user. An APAR fix is usually replaced later by a permanent correction called a PTF. APAR fixes are identified to SMP/E by the ++APAR statement.

**applied SYSMOD.** A SYSMOD that has been successfully processed by the SMP/E APPLY command. Applied SYSMODs do not have the ERROR flag set and are found as SYSMOD entries in the target zone.

**APPLY.** The SMP/E command used to install SYSMODs in the target libraries.

**apply.** In SMP/E, to install SYSMODs in the target libraries. This is done with the APPLY command.

**ASSEM entry.** An SMP/E entry containing assembler statements that can be assembled to create an object module.

**authorized program analysis report (APAR).** A report of a problem caused by a suspected defect in a current unaltered release of a program. The correction is called an APAR fix.

## B

**BACKUP entries.** A collection of SMP/E target zone entries that are copied into the SMPSCDS data set during APPLY processing before they are updated by inline JCLIN, a ++MOVE MCS, or a ++RENAME MCS, or before they are deleted by an element MCS with the DELETE operand.

BACKUP entries consist of:

- A SYSMOD entry indicating what entries have been added, deleted, or updated
- ASSEM entries for updated target zone ASSEM entries
- LMOD entries for updated target zone LMOD entries
- MAC entries for updated or deleted target zone MAC entries
- MOD entries for updated or deleted target zone MOD entries
- SRC entries for updated or deleted target zone SRC entries
- Data element entries for deleted target zone data element entries
- DLIB entries for updated target zone DLIB entries

**base function.** A SYSMOD defining elements of the base system or other products that were not previously present in the target libraries. Base functions are identified to SMP/E by the ++FUNCTION statement. SMP/E is an example of a base function.

**base level system.** The level of the target system modules, macros, source, and DLIBs created by system generation, to which function and service modifications

are applicable. OS/VS2 (MVS) Release 3.8 is an example of a base level system.

**base version of a load module.** Some load modules include modules both explicitly (through INCLUDE statements) and implicitly (through a SYSLIB allocation). The base version of such a load module includes only the explicitly-defined modules for the load module. It is maintained by SMP/E in the SMPLTS data set. The base version of a load module is used with the SYSLIB allocation as input to the link-edit utility in order to build the load module in its target libraries.

**binder.** A program that processes the output of language translators and compilers into an executable program (load module). Part of the DFSMS/MVS\* product or the MVS/DFP\* product, it replaces the link-edit utility and batch loader.

**bypass.** In SMP/E, to circumvent errors that would otherwise cause SYSMOD processing to fail. This is done by using the BYPASS operand on an SMP/E command.

## C

**causer SYSMOD.** A SYSMOD identified by root cause analysis to be at the base of errors that caused other SYSMODs to fail. See *root cause analysis*.

**CBIPO.** MVS Custom-Built Installation Process Offering.

**CBIPO driver.** A load-and-go MVS system designed to help users install a CBIPO if there is no MVS system to use as a driver.

**CBIPO Process Aids.** RIMs for a model MVS system, without any product code.

**CBIPO service tape.** A tape containing SYSMODs associated with the products in a CBIPO but not integrated into the CBIPO distribution libraries.

**CBPDO.** MVS Custom-Built Product Delivery Offering.

**CDS.** Control data set.

**control data set (CDS).** An SMP4 data set used to record information about a target library.

**CICS.** Customer Information Control System.

**CLEANUP.** The SMP/E command used to delete entries from the SMPMTS, SMPSTS, and SMPSCDS data sets after a SYSMOD has been accepted into the related distribution zone.

**CNTL.** See *SMPCNTL*.

**coexisting functions.** Functions that meet these requirements: (1) they are for the same system or subsystem and have the same SREL value, (2) they do not delete or supersede each other and are not negative prerequisites, and (3) if they are base functions, they are for different products. See also *conditionally coexisting functions* and *unconditionally coexisting functions*.

**conditional requisite queue (CRQ).** An SMP4 data set used to record information about a target library.

**conditional requisites.** Requisites defined by an ++IF statement. These are requisites that must be installed if the functions specified on the ++IF statements are installed.

**conditionally coexisting functions.** Functions that coexist but do not have to be in the same zone.

**consolidated software inventory.** See *SMPCSI*.

**CONVERT.** The SMP/E command used to move records from data sets for a previous release of SMP/E or SMP4 into data sets for the latest release of SMP/E.

**convert.** In SMP/E, to change records from SMP/E or SMP4 data sets into the format required for the latest release of SMP/E. This is done with the CONVERT command.

**corequisite SYSMODs.** SYSMODs each of which can be installed properly only if the other is present. Corequisites are defined by the REQ operand on the ++VER statement.

**corrective service.** Any SYSMOD used to selectively fix a system problem. Generally, corrective service refers to APAR fixes.

**cross-zone.** (1) A target zone other than the set-to zone that defines a load module containing modules from set-to zone. Also called a *TIEDTO zone*. The modules were added to the load module through the SMP/E LINK command. The relationship between the cross-zone and the set-to zone is established through the TIEDTO subentry in their zone definition entries. See also *set-to zone* and *TIEDTO relationship*. (2) Pertaining to relationships between zones, especially as a result of conditional requisites (++IF statements) or LINK processing. See also *cross-zone requisite*, *cross-zone load module*, and *cross-zone module*.

**cross-zone load module.** A load module containing modules from a different zone as a result of LINK processing.

**cross-zone module.** A module included in a load module from a different zone as a result of LINK processing.

**cross-zone requisite.** A conditional requisite that must be installed in one zone because of another SYSMOD that is installed in a different zone. The REPORT command can be used to check information saved from ++IF statements and determine where any cross-zone requisites should be installed.

**CRQ.** Conditional requisite queue.

**CSI.** Consolidated software inventory data set. See *SMPCSI*.

**CSSF.** Customer Software Support Facility.

**CUM tape.** Cumulative service tape.

**cumulative service tape (CUM tape).** The tape sent with an order for a new function that contains all the current PTFs for that function.

**Customer Software Support Facility (CSSF).** A database in Information/Access that can be used to research APARs and to obtain information about preventive service planning.

## D

**data element.** An element that is not a macro, module, or source—for example, a dialog panel or sample code.

**DDDEF entry.** An SMP/E entry containing the information SMP/E needs in order to dynamically allocate a particular data set.

**DEBUG.** The SMP/E command used to obtain additional information for problem determination—for example, to trace messages or take dumps.

**debug.** In SMP/E, to obtain additional information for problem determination—for example, to trace messages or take dumps. This is done with the DEBUG command.

**deleted function.** In SMP/E, a function that was removed from the system when another function was installed. This is indicated by the DELBY subentry in the SYSMOD entry for the deleted function. See also *explicitly deleted function* and *implicitly deleted function*.

**deleting function.** A function that removes other functions from the system. This is done by specifying them on the DELETE operand of the ++VER statement.

**dependent function.** A function that introduces new elements or redefines elements of the base level system or other products. A dependent function cannot exist without a base function. Dependent functions are identified to SMP/E by the ++FUNCTION statement.

**DFP.** Data Facility Product.

**DFSMS environment.** An environment that helps automate and centralize the management of storage. This is achieved through a combination of hardware, software, and policies. In the DFSMS environment for MVS, this function is provided by MVS/ESA SP and DFSMS/MVS or MVS/DFP, DFSORT, and RACF.

**dialog.** The interactive support provided by SMP/E through ISPF. Instead of entering specific commands and operands, you can use panels to specify the desired processing.

**distribution library (DLIB).** A library that contains the master copy of all the elements in a system. A distribution library can be used to create or back up a target library.

**distribution zone.** In SMP/E, a group of records in a CSI data set that describes the SYSMODs and elements in a distribution library.

**DLIB.** Distribution library.

**DLIB entry.** An SMP/E entry describing a distribution library that has been totally copied into a target library.

**DLIBZONE entry.** An SMP/E entry containing information used by SMP/E to process a specific distribution zone and the associated distribution libraries.

## E

**EC.** Engineering change.

**element.** In SMP/E, part of a product, such as a macro, module, dialog panel, or sample code.

**element MCS.** An MCS used to replace or update an element.

**element selection.** The process by which SMP/E chooses the appropriate changes for an element affected by several SYSMODs being installed at the same time.

**entry.** In SMP/E, a collection of records in a CSI data set. An entry can be created, updated, or deleted by use of UCL statements.

**environment.** The functions (FMIDs) installed on a particular system or subsystem (SREL).

**ERROR indicator.** In SMP/E, an indicator in a target or distribution zone SYSMOD entry that shows that SYSMOD processing failed. The ERROR indicator is set before SMP/E updates any libraries and is reset if

processing is successful. If processing fails, it remains set to show that an error occurred.

**ESO.** Expanded service options.

**exception SYSMOD.** A SYSMOD that is in error or that requires special processing before it can be installed. ++HOLD and ++RELEASE statements identify exception SYSMODs.

**expanded service options (ESO).** A tape that includes preventive service PTFs. Where available, it replaces PUTs as the vehicle for delivering preventive service. An ESO contains PTFs and ++ASSIGN statements assigning source IDs for the PTFs. In the United States, this tape is available from the IBM Support Center and can be ordered either by subscription or as needed.

**explicitly deleted function.** A function deleted because it was specified on the DELETE operand of a ++VER statement in another SYSMOD.

**exported zone.** A zone copied into a sequential data set by use of the SMP/E ZONEEXPORT command.

**external HOLDDATA.** ++HOLD statements contained in SMPHOLD. Contrast with *internal HOLDDATA*.

## F

**FE.** Field engineering.

**feature.** See *dependent function*.

**FMID.** Function modification identifier.

**FMIDSET.** A group of FMIDs to be used in processing an SMP/E command—for example, to indicate that SYSMODs applicable to certain functions should be installed.

**FMIDSET entry.** An SMP/E entry defining an FMIDSET.

**function.** In SMP/E, a product (such as a system component or licensed program) that can be installed in a user's system if desired. Functions are identified to SMP/E by the ++FUNCTION statement. Each function must have a unique FMID.

**function modification identifier (FMID).** The SYSMOD ID of a function SYSMOD. It identifies the function that currently owns a given element.

**functionally higher SYSMOD.** A SYSMOD that uses the function contained in an earlier SYSMOD (called the *functionally lower SYSMOD*) and contains additional functions as well.



**functionally lower SYSMOD.** A SYSMOD whose function is also contained in a later SYSMOD (called the *functionally higher SYSMOD*).

## G

**GENASM.** A subentry in the MAC entry that lists the ASSEM or SRC entries that must be assembled if the macro is replaced or updated.

**GENERATE.** The SMP/E command used to create a job stream that builds a set of target libraries from a set of distribution libraries.

**generate.** In SMP/E, to create a job stream that builds a set of target libraries from a set of distribution libraries. This is done with the GENERATE command.

**global zone.** A group of records in a CSI data set used to record information about SYSMODs received for a particular system. The global zone also contains information that (1) enables SMP/E to access target and distribution zones in that system, and (2) enables you to tailor aspects of SMP/E processing.

**GLOBALZONE entry.** An SMP/E entry containing information that SMP/E uses to process the global zone, the associated target and distribution zones, and the SMPPTS data set.

## H

**header MCS.** An ++APAR, ++FUNCTION, ++PTF, or ++USERMOD statement. The header MCS indicates the type of SYSMOD.

**HFS.** Hierarchical file system.

**HFS element.** An element that has a hierarchical file system as its “target library.”

**hierarchy.** In SMP/E, the top-down structure of function and service SYSMODs, in which each SYSMOD is dependent on the one above it.

**higher functional level.** An element version that contains all the functions of all other relevant versions of that element.

**HOLDDATA.** In SMP/E, MCSs used to indicate that certain SYSMODs contain errors or require special processing before they can be installed. ++HOLD and ++RELEASE statements are used to define HOLDDATA. SYSMODs affected by HOLDDATA are called *exception SYSMODs*.

**HOLDDATA entry.** An SMP/E entry containing ++HOLD statements that either were received from

SMPHOLD (external HOLDDATA) or were within a SYSMOD that was received (internal HOLDDATA).

## I

**ICF.** Integrated Catalog Facility.

**IFREQ.** A conditional requisite. Conditional requisites are specified on the REQ operand of the ++IF statement.

**IMASPZAP.** The system utility program used to install superzaps, which are changes for modules, load modules, or CSECTs within modules.

**implicitly deleted function.** A function deleted because of its dependency on an explicitly deleted function that is specified on the DELETE operand of the ++VER statement.

**imported zone.** A zone copied from a sequential data set into another zone by use of the SMP/E ZONEIMPORT command.

**IMS.** Information Management System.

**IMSGEN.** IMS generation.

**indicator.** See *subentry indicator*.

**in effect.** Having control over SMP/E processing. For example, an OPTIONS entry is in effect if (1) it is specified on the SET command or (2) it is defined as the default OPTIONS entry for the set-to zone.

**Information/Access.** A feature of Information/System 1, an interactive retrieval program. Information/Access provides direct customer access to a database that contains APAR and PTF information (CSSF), APAR fixes and PTFs, and preventive service planning information.

**inline data.** Information (such as utility control statements or code for an element) that is packaged directly after the associated MCS, rather than in a separate file or data set.

**inline JCLIN.** The JCL statements associated with a ++JCLIN statement. Inline JCLIN may immediately follow the ++JCLIN statement, or it may be in the RELFILE or TXLIB data set pointed to by the ++JCLIN statement. Inline JCLIN is used to update the target zone when a SYSMOD is applied, or the distribution zone when a SYSMOD is accepted. Contrast with *JCLIN input*.

**inner macro.** A macro invoked by another macro. In particular, inner macros are those that SMP/E does not detect during JCLIN processing of assembler job steps.

**install.** In SMP/E, to apply a SYSMOD to the target libraries or to accept a SYSMOD into the distribution libraries.

**internal HOLDDATA.** ++HOLD statements contained within a SYSMOD. Contrast with *external HOLDDATA*.

**I/O.** Input or output.

**IOGEN.** Input/output device generation.

**IPL.** Initial program load.

**ISMD.** IBM Software Manufacturing and Delivery (formerly called *PID*).

**ISPF.** Interactive System Productivity Facility.

**ISPF/PDF.** Interactive System Productivity Facility/Program Development Facility.

**IVP.** Installation verification procedure.

## J

**JCL.** Job control language.

**JCLIN.** (1) The SMP/E command used to process data from the SMPJCLIN data set. (2) The ++JCLIN statement, which is associated with JCLIN data that is included in a SYSMOD. (3) The SMPJCLIN data set. See *SMPJCLIN*.

See also *inline JCLIN* and *JCLIN data*.

**JCLIN data.** The JCL statements associated with the ++JCLIN statement or saved in the SMPJCLIN data set. They are used by SMP/E to update the target zone when the SYSMOD is applied. Optionally, SMP/E can use JCLIN data to update the distribution zone when the SYSMOD is accepted.

**JCLIN input.** The JCL statements contained in the SMPJCLIN data set and used as input for the JCLIN command. Contrast with *inline JCLIN*.

**job control language (JCL).** A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

## L

**licensed program.** A program that performs a function for the user, and usually interacts with and relies upon the system control program or some other IBM-provided control program. Generally, a licensed program is a software package that can be ordered from the program

libraries, such as IBM Software Distribution (ISMD). IMS and CICS are examples of licensed programs.

**link library (LKLIB).** A data set containing link-edited object modules.

**LIST.** The SMP/E command used to display entries in SMP/E data sets.

**list.** In SMP/E, to display entries in SMP/E data sets. This is done with the LIST command.

**LKLIB.** Link library.

**LMOD.** In SMP/E, an abbreviation for load module.

**LMOD entry.** An SMP/E entry containing all the information needed to replace or update a given load module.

**load module.** A computer program in a form suitable for loading into main storage for execution. It is usually the output of a link-edit utility.

**LOG.** (1) The SMP/E command used to write user-supplied information to the SMPLOG data set. (2) The SMPLOG data set. See *SMPLOG*.

**lower functional level.** An element version that is contained in a later element version.

## M

**MAC.** The SMP/E entry or MCS that describes a macro.

**macro.** An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language.

**MACUPD.** The SMP/E MCS used to update a macro.

**mass-mode processing.** In SMP/E, processing that includes all eligible SYSMODs, regardless of whether they were individually selected.

**master CSI.** The CSI data set that contains the global zone.

**MCS.** Modification control statement.

**MCS entry.** An SMP/E entry containing a copy of a SYSMOD exactly as it was received from the SMPPTFIN data set. MCS entries are in the SMPPTS data set, which is used as a warehouse for SYSMODs.

**MOD.** The SMP/E entry or MCS that describes an object module or a single-module load module.

**MODID.** Modification identifier.

**modification.** In SMP/E, an alteration or correction to a system control program, licensed program, or user program. Synonymous with *system modification* (SYSMOD).

**modification control statement (MCS).** An SMP/E control statement used to package a SYSMOD. MCSs describe the elements of a program and the relationships that program has with other programs that may be installed on the same system.

**modification identifier (MODID).** A list of SYSMOD IDs, including the last SYSMOD that totally replaced the element (RMID), any subsequent partial updates to the element (UMIDs), and the function that owns the element (FMID). MODIDs are contained in element entries.

**modification level.** A distribution of all temporary fixes that have been issued since the previous modification level. A change in modification level does not add new functions or change the programming support category of the release to which it applies. Contrast with *release* and *version*.

**Note:** Whenever a new release of a program is shipped, the modification level is set to 0. When the release is reshipped with the accumulated services changes incorporated, the modification level is incremented by 1.

**module.** Synonym for *object module* or *single-module load module*.

**MTS.** Macro temporary storage data set. See *SMPMTS*.

**MTSMAC entry.** An SMP/E entry that is a copy of a macro that resides only in a distribution library but is needed temporarily during APPLY processing. MTSMAC entries are in the SMPMTS data set.

**MVS.** Multiple Virtual Storage.

**MVS Custom-Built Installation Process Offering (CBIPO).** A software delivery offering used to create or replace an entire MVS, NCP, CICS, or IMS system.

**MVS Custom-Built Product Delivery Offering (CBPDO).** A software delivery offering used to add products or service to an existing MVS, NCP, CICS, or IMS system.

**MVS/ESA.** Multiple Virtual Storage/Enterprise Systems Architecture (MVS/SP\* Version 3).

**MVS/SP.** Multiple Virtual Storage/System Product.

**MVS/XA.** Multiple Virtual Storage/Extended Architecture (MVS/SP Version 2).

**MVS/370.** Multiple Virtual Storage for System/370\* (MVS/SP Version 1).

## N

**NCP.** Network Control Program.

**negative prerequisite (NPRE).** In SMP/E, a function that is mutually exclusive with another function. It is defined by the NPRE operand on the ++VER statement.

**NPRE.** Negative prerequisite.

## O

**object deck.** Object module input to the link-edit utility that is placed in the input stream, in card format.

**object module.** A module that is the output from a language translator (such as a compiler or an assembler). An object module is in relocatable format with machine code that is not executable. Before an object module can be executed, it must be processed by the link-edit utility.

When an object module is link-edited, a load module is created. Several modules can be link-edited together to create one load module (for example, as part of SMP/E APPLY processing), or an object module can be link-edited by itself to create a single-module load module (for example, to prepare the module for shipment in RELFILE format or in an LKLIB data set or as part of SMP/E ACCEPT processing).

**operating system.** In SMP/E, the system updated by APPLY and RESTORE processing. It consists of the target libraries. Also called the target system.

**OPTIONS entry.** An SMP/E entry defining processing options that are to be used by SMP/E.

**OS/VS.** Operating System/Virtual Storage (VS1 or VS2).

**OS/VS1.** Operating System/Virtual Storage 1.

**OS/VS2.** Operating System/Virtual Storage 2.

## P

**packaging.** Adding the appropriate MCS statements to elements to create a SYSMOD, then putting the SYSMOD in the proper format on the distribution medium, such as a tape or direct access data sets.

**PARMLIB.** The data set that contains members to

define parameters, such as macros and assembler operation codes.

**partitioned data set extended (PDSE).** A system-managed data set containing an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

**PE.** See *program error PTF*.

**PE-PTF.** See *program error PTF*.

**PID.** The former name for ISD.

**PRE.** Prerequisite.

**prerequisite (PRE).** In SMP/E, a SYSMOD that must be installed before or along with another SYSMOD in order for that other SYSMOD to be successfully installed. It is defined by the PRE operand on the ++VER statement.

**preventive service.** (1) The mass installation of PTFs to avoid rediscoveries of the APARs fixed by those PTFs. (2) The SYSMODs delivered on the program update tape.

**preventive service planning (PSP).** Installation recommendations and HOLDDATA for a product or a service level. PSP information can be obtained through CSSF or the IBM Support Center.

**product.** Generally, a software package, such as a licensed program or a user application. A product can contain one or more functions and can consist of one or more versions and releases.

**product version.** All the releases for a given version of a product.

**program error PTF (PE-PTF).** A PTF that has been found to contain an error. A PE-PTF is identified on a ++HOLD ERROR statement, along with the APAR that first reported the error.

**program object.** An executable program stored in a PDSE program library. It is similar to a load module, but has fewer restrictions. For SMP/E purposes, program objects are referred to as load modules.

**program packaging.** See *packaging*.

**program product.** The former term for licensed program.

**program temporary fix (PTF).** A temporary solution or bypass of a problem that may affect all users and that was diagnosed as the result of a defect in a current

unaltered release of the program. In the absence of a new release of a system or component that incorporates the correction, the fix is not temporary but is the permanent and official correction mechanism. New elements can also be defined in a PTF. PTFs are identified to SMP/E by the ++PTF statement.

**program update tape (PUT).** The former vehicle for preventive service. See *expanded service options*.

**PSP.** Preventive service planning.

**PTF.** Program temporary fix.

**PTS.** PTF temporary store data set. See *SMPPTS*.

**PTFIN.** PTF input data set. See *SMPPTFIN*.

**PUT.** See *expanded service options*.

## R

**RACF.** Resource Access Control Facility.

**RECEIVE.** The SMP/E command used to read in SYSMODs and other data from the SMPPTFIN and SMPHOLD data sets.

**receive.** In SMP/E, to read SYSMODs and other data from the SMPPTFIN and SMPHOLD data sets, and store them on the global zone for subsequent SMP/E processing. This is done with the RECEIVE command.

**regressed SYSMOD.** A SYSMOD one or more of whose elements are modified by subsequent SYSMODs that are not related to it.

**regressing SYSMOD.** A SYSMOD that causes regression of previous modifications when it is installed.

**regression.** In SMP/E, the condition that occurs when an element is changed by a SYSMOD that is not related to SYSMODs that previously modified the element.

**REJECT.** The SMP/E command used to remove SYSMODs from the global zone and the SMPPTS data set.

**reject.** In SMP/E, to remove SYSMODs from the global zone and SMPPTS and delete any related SMPTLIB data sets. This is done with the REJECT command.

**related installation materials (RIMs).** In CBIPO, task-oriented documentation, jobs, sample exit routines, procedures, parameters, and examples developed by IBM. RIMs are designed to help users install, generate, and use systems created from a CBIPO.

**related SYSMOD.** A SYSMOD associated with other SYSMODs by the FMID, PRE, REQ, or SUP operands.

**related zone.** The zone named in the RELATED sub-entry of a TARGETZONE or DLIBZONE entry. For a target zone, the related zone is generally the distribution zone for the libraries used to create the target libraries. For a distribution zone, the related zone is generally the target zone for the libraries built from the distribution libraries.

**relative file (RELFILE) format.** A SYSMOD packaging method where elements and JCLIN data are in separate relative files from the MCSs. When SYSMODs are packaged in relative file format there is a file of MCSs for one or more SYSMODs, and one or more relative files containing unloaded source-code data sets and unloaded link-edited data sets containing executable modules. The relative files can be either unloaded files in IEBCOPY format, or they can be partitioned data sets. Relative file format is the typical method used for packaging function SYSMODs.

**relative files (RELFILEs).** Unloaded files containing modification text and JCL input data associated with a SYSMOD. These files are used to package a SYSMOD in relative file format.

**release.** A distribution of a new product or new function and APAR fixes for an existing product. Contrast with *modification level* and *version*.

**replacement modification identifier (RMID).** The SYSMOD ID of the last SYSMOD that completely replaced a given element.

**REPORT.** The SMP/E command used to obtain information about SYSMODs that have been installed. These are the types of REPORT commands:

- REPORT CALLLIBS: Identifies load modules that need to be relinked because implicitly-included modules in a particular library have been updated.
- REPORT CROSSZONE: Lists conditional requisites that must be installed in certain zones because of SYSMODs installed in other zones.
- REPORT ERRSYSMODS: Determines whether any SYSMODs already installed are now exception SYSMODs.
- REPORT SOURCEID: Lists the source IDs associated with SYSMODs in the specified zones.
- REPORT SYSMODS: Compares the SYSMODs installed in two target or distribution zones.

**requisite.** A SYSMOD that must be installed before or at the same time as the SYSMOD being processed. There are several types of requisites:

- Prerequisites, which are specified by the PRE operand on the SYSMOD's ++VER statement
- Corequisites, which are specified by the REQ operand on the SYSMOD's ++VER statement
- Conditional requisites, which are specified by the REQ operand on the SYSMOD's associated ++IF statement

**RESETRC.** The SMP/E command used to set the return codes for the previous commands to zero, so that SMP/E can process the current command.

**RESTORE.** The SMP/E command used to remove applied SYSMODs from the target libraries.

**restore.** In SMP/E, to remove applied SYSMODs from the target libraries by use of the RESTORE command.

**restore group.** All the SYSMODs that have a direct or indirect relationship with a SYSMOD being restored by use of the GROUP operand.

**RETAIN.** A database, accessible through Information/Access, that contains information about APARs and PTFs. The customer version of this database is called the Customer Service Support Facility (CSSF).

**RIM.** Related installation material.

**RMID.** Replacement modification identifier.

**root cause analysis.** Processing done by SMP/E for the ACCEPT, APPLY, and RESTORE commands to identify causer SYSMODs (SYSMODs whose failure has led to the failure of other SYSMODs). The types of errors SMP/E analyzes to determine causer SYSMODs include the following:

- Held SYSMODs
- Missing requisite SYSMODs
- Utility program failures: copy, update, assembler, link, zap
- Out-of-space conditions: x37 abends
- Missing DD statements and other allocation errors
- ID errors (a SYSMOD does not supersede or specify as a prerequisite an RMID or a UMID)
- JCLIN failures (syntax errors)

## S

**SCDS.** Save control data set. See *SMPSCDS*.

**SCP.** System control program.

**select-mode processing.** In SMP/E, processing that includes individually selected SYSMODs.

**SERV.** CBIPO service tape.

**service.** PTFs and APAR fixes.

**service level.** The FMID, RMID, and UMID values for an element. The service level identifies the owner of the element, the last SYSMOD to replace the element, and all the SYSMODs that have updated the element since it was last replaced.

**service order relationship.** A relationship among service SYSMODs that is determined by the PRE and SUP operands, and the type of SYSMOD.

**service SYSMOD.** Any SYSMOD identified by an ++APAR or ++PTF statement.

**service update.** The integration of available service into the current release of a function. Since this is not a new release of the function, it does not change the function's FMID.

**SET.** The SMP/E command used to indicate the zone to be processed.

**set.** In SMP/E, to indicate which zone should be processed by the subsequent commands. This is done with the SET command.

**set-to zone.** The zone that was specified on the previous SET command and that is currently being processed. Contrast with *cross-zone*.

**single-module load module.** A load module created by link-editing a single object module by itself—for example, to prepare the module for shipment in RELFILE format or in an LKLIB data set or as part of SMP/E ACCEPT processing.

**SMP.** System Modification Program.

**SMPCNTL.** The SMP/E data set that contains the SMP/E commands to be processed.

**SMPCSI.** The SMP/E data set that contains information about the structure of a user's system as well as information needed to install the operating system on a user's system. The SMPCSI DD statement refers specifically to the CSI that contains the global zone. This is also called the *master CSI*.

**SMPDEBUG.** The SMP/E data set that contains a dump requested by the DEBUG command. Depending on the operands specified, it may contain (1) a dump of SMP/E control blocks and storage areas associated with the specified dump points or (2) a dump of the VSAM RPL control block for the specified SMP/E function.

**SMP/E.** System Modification Program Extended.

**SMP/E commands.** Commands defining the processing to be done by SMP/E, such as RECEIVE.

**SMP/E entry.** An entry in an SMP/E data set—for example, a MOD entry in a CSI data set.

**SMPHOLD.** The SMP/E file or data set that contains HOLDDATA (++HOLD and ++RELEASE statements) to be processed by the RECEIVE command.

**SMPJCLIN.** The SMP/E data set that contains a job stream of assembly, link-edit, and copy job steps. This data is typically the stage 1 output from the most recent full or partial system generation. However, it may also be other data in a similar format, such as the output of the GENERATE command. This job stream is used as input to the JCLIN command to update or create entries in a target zone.

**SMPLIST.** The SMP/E data set that contains the output of all LIST commands.

**SMPLOG.** The SMP/E data set that contains time-stamped records of SMP/E processing. The records in this data set can be written automatically by SMP/E or added by the user through the LOG command.

**SMPLOGA.** A secondary log data set for SMP/E processing. If SMPLOGA is defined, it is automatically used when the SMPLOG data set is full.

**SMPLTS.** The SMP/E data set used as a target load module library to maintain the base version of a load module that specifies a SYSLIB allocation in order to implicitly include modules.

**SMPMTS.** The SMP/E data set used as a target library for macros that exist only in a distribution library, such as macros in SYS1.AMODGEN. The SMPMTS enables the current version of these macros to be used for assemblies during APPLY processing.

**SMPOBJ.** The SMP/E data set used for source-maintained products. SMPOBJ contains preassembled modules that can be used to avoid reassembling those modules. These modules must be in load module format—that is, in the same format as modules residing in the distribution library.

**SMPOUT.** The SMP/E data set that contains messages issued during SMP/E processing. It may also contain a dump of the VSAM RPL, if a dump was taken. In addition, it may contain LIST output and reports if the SMPLIST and SMPRPT data sets are not defined.

**SMPPTFIN.** The SMP/E file or data set that contains SYSMODs and ++ASSIGN statements to be processed by the RECEIVE command.

**SMPPTS.** The SMP/E data set that contains SYSMODs received from the SMPPTFIN data set. The SMPPTS is a sort of warehouse, and is the source of SYSMODs that are installed in the target and distribution libraries.

**SMPPUNCH.** The SMP/E data set that contains output from various SMP/E commands. This output generally consists of commands or control statements.

- **GENERATE:** A job stream for building target libraries
- **REPORT:** Commands for installing or listing SYSMODs
- **UNLOAD:** UCLIN statements for recreating the entries that were unloaded

**SMPRPT.** The SMP/E data set that contains the reports produced during SMP/E processing.

**SMPSCDS.** The SMP/E data set that contains backup copies of target zone entries that are created during APPLY processing. These backup copies are made before the entries are (1) changed by inline JCLIN, a ++MOVE MCS, or a ++RENAME MCS, or (2) deleted by an element MCS with the DELETE operand. The backup copies are used during RESTORE processing to return the entries to the way they were before APPLY processing.

**SMPSNAP.** The SMP/E data set that is used for snap dump output. When a severe error such as an abend or severe VSAM return code occurs, SMP/E requests a snap dump of its storage before doing any error recovery. In addition, the DEBUG command can request a snap dump of SMP/E storage when specified messages are issued, or can request a snap dump of control blocks and storage areas associated with a specified dump point.

**SMPSTS.** The SMP/E data set used as a target library for source that exists only in a distribution library. The SMPSTS enables the current version of this source to be used for assemblies during APPLY processing.

**SMPTLIB.** The SMP/E data sets used as temporary storage for relative files loaded from SMPPTFIN during RECEIVE processing. The SMPTLIB data sets are

deleted when the associated SYSMOD is deleted by REJECT, RESTORE, or ACCEPT processing.

**SMPWRK1.** The SMP/E data set used as temporary storage for macro updates and replacements that will be processed by an update or copy utility program. SMP/E places the input in SMPWRK1 during APPLY and ACCEPT processing before calling the utility.

**SMPWRK2.** The SMP/E data set used as temporary storage for source updates and source replacements that will be processed by an update or copy utility program. SMP/E places the input in SMPWRK2 during APPLY and ACCEPT processing before calling the utility.

**SMPWRK3.** The SMP/E data set used as temporary storage for object modules supplied by a SYSMOD, object modules created by assemblies, and zap utility input following ++ZAP statements.

**SMPWRK4.** The SMP/E data set used as temporary storage for zap utility or link-edit utility input that contains EXPAND control statements.

**SMPWRK6.** The SMP/E data set used during ACCEPT and APPLY processing as temporary storage for inline replacements for data elements. SMP/E places the input in this data set so that it can be directly accessed and installed by the copy utility or SMP/E.

**SMP4.** System Modification Program Release 4.

**source.** The source statements that constitute the input to a language translator for a particular translation.

**source ID.** A 1- to 8-character identifier that indicates how a SYSMOD was obtained—for example, from a particular service level in an ESO. A source ID is associated with a specific SYSMOD by the RECEIVE command or the ++ASSIGN statement.

**SOURCEID.** The operand used to refer to a source ID.

**source module.** The source statements that constitute the input to a language translator, such as a compiler or an assembler, for a particular translation.

**SRC.** The SMP/E entry or MCS statement that describes a source.

**SRCUPD.** The MCS used to update a source.

**SREL.** System release identifier.

**Storage Management Subsystem (SMS).** A DFSMS/MVS or MVS/DFP facility used to automate and centralize the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and

retention requirements, and storage requirements to the system through data class, storage class, management class, storage group, and ACS routine definitions.

**STS.** Source temporary store data set. See *SMPSTS*.

**STSSRC entry.** An SMP/E entry that is a copy of source that resides only in a distribution library but is needed temporarily during APPLY processing. STSSRC entries are in the SMPSTS data set.

**stub entry.** An element entry or LMOD entry that does not contain the basic information SMP/E requires in order to process the element or load module (such as FMID, RMID, or library names), but does contain other information, such as subentries describing cross-zone relationships.

**stub load module.** A load module that does not contain the modules needed to perform its basic functions, but does contain other modules, such as cross-zone modules.

**subentry.** A field in an SMP/E entry. Each subentry has associated with it a type and a value. The same subentry type may occur several times in a single entry, each time with a different value. For example, the modules supplied by a PTF are saved as MOD subentries in the PTF's SYSMOD entry. Some subentries occur only once within an entry, such as the FMID subentry in a target zone MOD entry.

**subentry indicator.** A subentry that does not have a data value associated with it. An example of an indicator is the ERROR indicator in the SYSMOD entry. An indicator is either on or off.

**subentry list.** Multiple occurrences of the same subentry type in an entry, each with a different value. For example, the modules supplied by a PTF are saved as names in the MOD subentry list within the SYSMOD entry for that PTF.

**SUP.** Supersede.

**superseded-only SYSMOD.** A SYSMOD that has not been installed, but that has been superseded by another SYSMOD that has been installed.

**superseded SYSMOD.** In SMP/E, a SYSMOD that is contained in or replaced by the SYSMOD or requisite set of SYSMODs currently being processed. This is indicated by the SUPBY subentry in the SYSMOD entry for the superseded SYSMOD. A superseded SYSMOD is functionally lower than the SYSMOD that superseded it.

**superseding SYSMOD.** In SMP/E, a SYSMOD that contains all the functions in another SYSMOD and is recognized as the equivalent of that other SYSMOD.

The superseding SYSMOD uses SUP operand on its ++VER statement to specify the superseded SYSMOD.

**superzap.** A generic term for the process performed by IMASPZAP. It can also refer to the module updates processed by IMASPZAP.

**SYSGEN.** System generation.

**SYSLIB.** (1) A subentry used to identify the target library in which an element is installed. (2) A concatenation of macro libraries to be used by the assembler. (3) A set of routines used by the link-edit utility to resolve unresolved external references.

**SYSMOD.** System modification.

**SYSMOD entry.** An SMP/E entry containing information about a SYSMOD that has been received into the SMPPTS, accepted into the distribution libraries, or applied to the target libraries.

**SYSMOD ID.** System modification identifier.

**SYSMOD packaging.** See *packaging*.

**SYSMOD selection.** The process of determining which SYSMODs are eligible to be processed.

**SYSPRINT.** The data set that contains output from the utilities called by SMP/E.

**SYSPUNCH.** The temporary data set containing object modules assembled by running the job stream produced by system generation or the GENERATE command. These modules are not installed in the distribution libraries at ACCEPT time.

**system control program (SCP).** IBM-supplied programming that is fundamental to the operation and maintenance of the system. It serves as an interface with licensed programs and user programs and is available without additional charge.

**system generation (SYSGEN).** The process of selecting optional parts of an operating system and of creating a particular operating system tailored to the requirements of a data processing installation.

**system modification (SYSMOD).** The input data to SMP or SMP/E that defines the introduction, replacement, or update of elements in the operating system and associated distribution libraries to be installed under the control of SMP or SMP/E. A system modification is defined by a set of MCS.

**system modification identifier (SYSMOD ID).** The name that SMP/E associates with a system modification. It is specified on the ++APAR, ++FUNCTION, ++PTF, or ++USERMOD statement.



**System Modification Program (SMP).** A program used to install software and software changes on OS/VS1 and OS/VS2 systems. In this book, SMP refers to SMP Release 4.

**System Modification Program Extended (SMP/E).** A licensed program used to install software and software changes on OS/VS1 and OS/VS2 systems. In addition to providing the services of SMP, SMP/E consolidates installation data, allows more flexibility in selecting changes to be installed, provides a dialog interface, and supports dynamic allocation of data sets. In this book, SMP/E refers to the current release of SMP/E and any earlier releases of SMP/E, unless specified otherwise.

**system release identifier (SREL).** A 4-byte value representing the system or subsystem, such as Z038 for MVS-based products.

**SYSUT1, SYSUT2, SYSUT3.** Scratch data sets for SMP/E and the utilities it calls.

**SYSUT4.** A data set that is used instead of the SYSIN data sets when certain utilities are called.

## T

**target library.** A library containing the executable code that makes up a system.

**target system.** The system updated during APPLY and RESTORE processing, also referred to as the operating system. See also target libraries.

**target zone.** In SMP/E, a group of records in a CSI data set that describes the SYSMODs, elements, and load modules in a target library.

**TARGETZONE entry.** An SMP/E entry containing information used by SMP/E to process a specific target zone and the associated target libraries.

**temporary data set.** A work data set (SMPWRK1–SMPWRK6) or utility data set (SYSUT1–SYSUT4). Temporary data sets are allocated when processing for an SMP/E command begins, and deleted when processing is finished.

**text library (TXLIB).** A data set containing JCLIN input or replacements for macros, source, or object modules that have not been link-edited. It is used when the JCLIN or elements are provided in partitioned data sets rather than inline or in relative files.

**TGTLIB.** Target library.

**TIEDTO relationship.** A cross-zone relationship between two target zones created when the LINK command updates a load module in one of the zones to

include modules from the other zone. This relationship is established through the TIEDTO subentry in the zone definition entries for each of the zones.

**TIEDTO zone.** See *cross-zone*.

**TLIB.** Temporary library. See *SMPTLIB*.

**transformed data.** Data processed by the GIMDTS service routine so that it can be packaged inline in fixed-block 80 records.

**TXLIB.** Text library.

## U

**UCL statement.** An SMP/E control statement used to define or change information in an SMP/E data set entry. UCL statements are coded between the UCLIN and ENDUCL commands. The UCL statement specifies the action to be taken (ADD, REP, or DEL), the entry to be modified, and any indicators and subentries to be changed.

**UCLIN.** The SMP/E command used to mark the beginning of UCL statements, which are used to make changes to entries in SMP/E data sets.

**UMID.** Update modification identifier.

**unconditionally coexisting functions.** Functions that coexist and must be in the same zone.

**UNLOAD.** The SMP/E command used to copy data out of SMP/E data set entries in the form of UCL statements.

**unload.** In SMP/E, to copy data out of SMP/E data set entries in the form of UCL statements, by use of the UNLOAD command.

**update.** In SMP/E, to change an existing element without replacing it.

**update modification identifier (UMID).** The SYSMOD ID of a SYSMOD that updated the last replacement of a given element.

**user modification (USERMOD).** A change constructed by a user to modify an existing function, add to an existing function, or add a user-defined function. USERMODs are identified to SMP/E by the ++USERMOD statement.

**USERMOD.** User modification.

**UTILITY entry.** An SMP/E entry containing information used by SMP/E to invoke a particular system utility program.

## V

**VERSION.** An operand on the ++VER or element statement. VERSION specifies one or more SYSMODs containing elements that are functionally lower than elements in the SYSMOD that specifies the operand. The VERSION operand is also used to change ownership of elements.

**version.** A separate licensed program that is based on an existing licensed program and that usually has significant new code or new functions. Contrast with *release* and *modification level*.

**versioned element.** An element that is part of more than one function—for example, one that is part of a base function and a dependent function.

**VSAM.** Virtual Storage Access Method.

**VS1.** Operating System/Virtual Storage 1.

**VS2.** Operating System/Virtual Storage 2.

**VTOC.** Volume table of contents.

## Z

**ZAP.** (1) The SMP/E MCS used to package an update for an object module. (2) The superzap control statement used to update an object module. (3) A shortened name for the superzap utility, which is used to install these updates. See *IMASPZAP*.

**zone.** A partition in a CSI data set.

**ZONECOPY.** The SMP/E command used to copy a zone from one CSI data set to another.

**ZONEDELETE.** The SMP/E command used to delete a zone from a CSI data set.

**ZONEEDIT.** The SMP/E command used to change the values for a subentry in all the DDDEF or UTILITY entries in a given zone.

**ZONEEXPORT.** The SMP/E command used to copy a zone into a sequential data set.

**ZONEIMPORT.** The SMP/E command used to load an exported zone from a sequential data set into another zone.

**ZONEMERGE.** The SMP/E command used to copy one zone into another, or to merge two zones into one.

**ZONERENAME.** The SMP/E command used to change the name of a zone.

**ZONESET.** A group of zones to be used when processing an SMP/E command. For example, it may define the zones that the REPORT command is to check for cross-zone requisites. A ZONESET may also define a group of zones to be checked or ignored by the REJECT command.

**ZONESET entry.** An SMP/E entry defining a ZONESET.

## Bibliography

This section tells you more about the SMP/E library, education on SMP/E, and additional publications you might find helpful.

### The SMP/E Library

The SMP/E Release 8.1 publications are available in hardcopy and BookManager\*-viewable softcopy.

- Table 46 lists the SMP/E Release 8.1 publications and briefly describes each one.

You can order binders and inserts for the hardcopy books. (Inserts are the slip-in covers and spine for a binder.) For information about binders and prepackaged groups of SMP/E publications, see Table 47 on page 896.

- Once SMP/E Release 8.1 is generally available, online versions of the SMP/E Release 8.1 books will be available on the subsequent edition of the IBM Online Library: MVS Base Collection Kit, CD-ROM **SK2T-0710**.

Table 46. Publications for SMP/E Release 8.1

Title	Description
<i>MVS Software Manufacturing Offerings General Information</i> , GC23-0351	Summarizes SMP/E, the CBIPO* offering, the CBPDO* offering, and the CustomPac offerings
<i>SMP/E R8.1 Primer</i> , GC23-3771	Introduces the basic principles needed for using SMP/E, without the expert-level details found in other SMP/E publications
<i>SMP/E R8.1 Master Index</i> , GC23-3812	Helps users quickly determine which book in the SMP/E library contains the information they are looking for
<i>SMP/E R8.1 Program Directory (English Feature)</i> , GC23-0130 <i>SMP/E R8.1 Program Directory (Japanese Feature)</i> , GC23-0469	Explains how to plan for installing SMP/E Release 8.1 with SMP/E Release 5 or higher
<i>SMP/E R8.1 User's Guide</i> , SC28-1302	Describes how to use SMP/E to install programs and service
<i>SMP/E R8.1 Messages and Codes</i> , SC28-1108	Explains SMP/E messages and return codes and the actions to take for each
<i>SMP/E R8.1 Reference</i> , SC28-1107	Explains SMP/E commands and processing in detail
<i>SMP/E R8.1 Reference Summary</i> , SX22-0006	Reviews the SMP/E commands in a convenient form
<i>SMP/E R8.1 CBIPO Dialogs User's Guide</i> , SC23-0538	Explains how to use the CBIPO dialogs to install, reinstall, and redistribute CBIPO orders
<i>Migrating to the CBIPO Dialogs</i> , GC23-3810	Compares functions provided in the batch CBIPO installation method with functions provided in the CBIPO dialogs contained in SMP/E
<i>SMP/E R8.1 Diagnosis Guide</i> , SC23-3130	Explains how to handle suspected SMP/E problems
<i>Standard Packaging Rules for MVS-Based Products</i> , SC23-3695	Explains how to package programs for installation by SMP/E

Item	Books Only	Books Plus All Necessary Binders and Binder Inserts	One Set of Binders and Inserts (No Books)	One Set of Binder Inserts (No Books, No Binders)
Complete SMP/E library	SBOF-1587 (English feature) or SBOF-3161 (Japanese feature)	SBOF-3163	SBOF-2136 (order 1) SBOF-2137 (order 2)	SX23-0442 (order 1) SX23-0443 (order 2)
SMP/E R8.1 Reference only	No BOF available	SBOF-1270	SBOF-2136	SX23-0442
Complete SMP/E library minus the SMP/E R8.1 Reference	No BOF available	SBOF-1271	SBOF-2137 (order 2)	SX23-0443 (order 2)
<b>Binder Only:</b> To get just a binder, with no inserts or manuals, order SX80-0258.				

Table 48 shows where to find the information necessary to perform various SMP/E tasks.

Where to Look	Evaluation	Planning	Installing SMP/E	Using SMP/E	Conversion	Diagnosis	Packaging
Help panels				X			
MVS Software Manufacturing Offerings General Information	X	X					
MVS Custom-Built Offerings Planning and Installation		X					
SMP/E R8.1 CBIPO Dialogs User's Guide		X		X			
Migrating to the CBIPO Dialogs		X		X			
SMP/E R8.1 Diagnosis Guide						X	
SMP/E R8.1 LPS	X						
SMP/E R8.1 Master Index				X			
SMP/E R8.1 Messages and Codes				X		X	
SMP/E R8.1 Primer				X			
SMP/E R8.1 Program Directory (English Feature) SMP/E R8.1 Program Directory (Japanese Feature)		X					

Table 48 (Page 2 of 2). Where to Find Information That Supports SMP/E Tasks

Where to Look	Evaluation	Planning	Installing SMP/E	Using SMP/E	Conversion	Diagnosis	Packaging
SMP/E Program Directory shipped with the product (one for the English feature, one for the Japanese feature)		X	X				
<i>Standard Packaging Rules for MVS-Based Products</i>		X					X
<i>SMP/E R8.1 Reference</i>		X		X	X	X	X
<i>SMP/E R8.1 Reference Summary</i>				X			
<i>SMP/E R8.1 User's Guide</i>		X		X			X
<i>What's New in SMP/E Release 8.1</i>	X						

## Classes and Self-Study Courses for SMP/E

Table 49 shows the recommended education on CBIPOs, CBPDOs, and SMP/E that is offered through the various IBM locations.

Table 49 (Page 1 of 2). Classes and Self-Study Courses

Location	Recommended Education	Catalog of Courses	Phone Number for More Information
Australia	<ul style="list-style-type: none"> <li>“SMP/E: A Guide for the New SMP/E User” (Self-Study Course 32186)</li> <li>“SMP/E Fundamentals” (Course H3765)</li> <li>“Integrated System Maintenance Using SMP/E” (Course H3763)</li> <li>“MVS Installation and Tailoring” (Course H3903)</li> <li>All prerequisites for the above classes, or equivalent experience</li> </ul>	Contact your local branch office.	Contact your local branch office.
Canada	<ul style="list-style-type: none"> <li>“SMP/E: A Guide for the New SMP/E User” (Self-Study Course 32186)</li> <li>“New SMP Users” (Course S4716)</li> <li>“MVS Installation and Tailoring” (Course S6375)</li> <li>All prerequisites for the above classes, or equivalent experience</li> </ul>	<i>Education Course Catalogue</i> , G209-0073 (bilingual version) or G209-0062 (English version)	IBM Direct-Education at 1-800-465-1234

Table 49 (Page 2 of 2). Classes and Self-Study Courses			
Location	Recommended Education	Catalog of Courses	Phone Number for More Information
EMEA	<ul style="list-style-type: none"> <li>• “SMP/E: A Guide for the New SMP/E User” (Self-Study Course 32186)</li> <li>• “System Installation and Maintenance with SMP/E”</li> <li>• “MVS/XA Installation Practice and Procedure” or “MVS/ESA Installation and Implementation”</li> <li>• “MVS/ESA Customization”</li> <li>• All prerequisites for the above classes, or equivalent experience</li> </ul>	See your country's education course catalog.	See your country's education course catalog for enrollment procedures.
Japan	<ul style="list-style-type: none"> <li>• “How to Use SMP/E” (Self-Study Course 25024)</li> <li>• “MVS Installation and Tailoring” (Course H3903)</li> <li>• “MVS/ESA Installation” (Course 24226)</li> <li>• “MVS/ESA Customization” (Course 24228)</li> <li>• All prerequisites for the above classes, or equivalent experience</li> </ul>	<i>Catalog of IBM Education</i> , GR18-5200	IBM DIRECT at 03-865-5748
United States	<ul style="list-style-type: none"> <li>• “SMP/E: A Guide for the New SMP/E User” (Self-Study Course 32186)</li> <li>• “SMP/E Fundamentals” (Course H3765)</li> <li>• “Integrated System Maintenance Using SMP/E” (Course H3763)</li> <li>• “MVS Installation and Tailoring” (Course H3903)</li> <li>• All prerequisites for the above classes, or equivalent experience</li> </ul>	<i>Catalog of IBM Education</i> , G320-1244	IBM DIRECT at 1-800-IBM-TEACH

## Related Publications

This section lists IBM manuals you may find useful when using SMP/E.

## General Interest

- *MVS Custom-Built Offerings Planning and Installation*, SC23-0352
- *MVS Software Manufacturing Offerings General Information*, GC23-0351
- *MVS Software Management Cookbook*, GG24-3481

## ISPF Version 4

- *ISPF V4.1 Dialog Developer's Guide and Reference*, SC34-4486
- *ISPF V4.1 Dialog Tag Language Guide and Reference*, SC34-4441
- *ISPF V4.1 Examples*, SC34-4451

## ISPF and ISPF/PDF Version 3

- *ISPF V3 MVS Dialog Management Guide*, SC34-4213
- *ISPF V3 MVS Dialog Management Services Examples*, SC34-4215
- *ISPF/PDF V3 MVS Guide*, SC34-4135

## ISPF and ISPF/PDF Version 2 Release 3

- *ISPF and PDF General Information*, GC34-4116
- *ISPF Dialog Management Guide*, SC34-4112
- *ISPF Dialog Management Services Examples*, SC34-4113
- *ISPF/PDF Guide*, SC34-4118

## OpenEdition MVS

- *Introducing OpenEdition MVS*, GC23-3010
- *MVS/ESA OpenEdition MVS Command Reference*, SC23-3013

## MVS/ESA SP Version 5

- *DFSMS Storage Administration Guide*, SC26-4929
- *MVS/ESA SP V5 HCD: Planning*, GC28-1445
- *MVS/ESA SP V5 HCD: User's Guide*, GC28-1445
- *MVS/ESA SP V5 JES2 Initialization and Tuning Guide*, SC28-1453
- *MVS/ESA SP V5 JES2 Initialization and Tuning Reference*, SC28-1454
- *MVS/ESA SP V5 JES3 Initialization and Tuning Guide*, SC28-1455
- *MVS/ESA SP V5 JES3 Initialization and Tuning Reference*, SC28-1456
- *MVS/ESA SP V5 Planning: Global Resource Serialization*, GC28-1450
- *MVS/ESA SP V5 Setting Up a Sysplex*, GC28-1449
- *MVS/ESA SP V5 System Data Set Definition*, GC28-1432
- *MVS/ESA SP V5 System Messages, Volume 1 (ABA-ASA)*, GC28-1480
- *MVS/ESA SP V5 System Messages, Volume 2 (ASB-ERB)*, GC28-1480
- *MVS/ESA SP V5 System Messages, Volume 3 (GFSA-IEB)*, GC28-1480
- *MVS/ESA SP V5 System Messages, Volume 4 (IEC-IFD)*, GC28-1480
- *MVS/ESA SP V5 System Messages, Volume 5 (IGD-IZP)*, GC28-1480
- *Sysplex Overview*, GC28-1208

## MVS/ESA SP Version 4

- *DFSMS/MVS Program Management*, SC26-4916
- *MVS/ESA Library Guide*, GC28-1601
- *MVS/ESA Application Development Guide: Authorized Assembler Language Programs*, GC28-1645
- *MVS/ESA Application Development Guide: Batch Local Shared Resources*, GC28-1672
- *MVS/ESA Data Areas, Volume 1*, LY28-1821
- *MVS/ESA Data Areas, Volume 2*, LY28-1822
- *MVS/ESA Data Areas, Volume 3*, LY28-1823
- *MVS/ESA Data Areas, Volume 4*, LY28-1824
- *MVS/ESA Data Areas, Volume 5*, LY28-1825
- *MVS/ESA Hardware Configuration Definition: Using the Dialog*, GC33-6457
- *MVS/ESA Initialization and Tuning Guide*, GC28-1634

- *MVS/ESA Initialization and Tuning Reference*, GC28-1635
- *MVS/ESA MVS Configuration Program*, GC28-1615
- *MVS/ESA Planning: Problem Determination and Recovery*, GC28-1629
- *MVS/ESA Problem Determination Guide*, GC28-1667
- *MVS/ESA Service Aids*, GC28-1669
- *MVS/ESA System Data Set Definition*, GC28-1612
- *MVS/ESA System Messages, Volume 1*, GC28-1656
- *MVS/ESA System Messages, Volume 2*, GC28-1657
- *MVS/ESA System Messages, Volume 3*, GC28-1658

## MVS/ESA (MVS/SP Version 3)

- *MVS/ESA Library User's Guide*, GC28-1563
- *MVS/ESA Application Development Guide: Authorized Assembler Language Programs*, GC28-1645
- *MVS/ESA Basics of Problem Determination*, GC28-1839
- *MVS/ESA Batch Local Shared Resources Subsystem*, GC28-1059
- *MVS/ESA Catalog Administration Guide*, SC26-4502
- *MVS/ESA Configuration Program Guide and Reference*, GC28-1817
- *MVS/ESA Data Administration Guide*, SC26-4505
- *MVS/ESA Data Administration: Macro Instruction Reference*, SC26-4517
- *MVS/ESA Data Administration: Utilities*, SC26-4516
- *MVS/ESA Data Areas, Volume 1*, LY28-1821
- *MVS/ESA Data Areas, Volume 2*, LY28-1822
- *MVS/ESA Data Areas, Volume 3*, LY28-1823
- *MVS/ESA Data Areas, Volume 4*, LY28-1824
- *MVS/ESA Data Areas, Volume 5*, LY28-1825
- *MVS/ESA Diagnosis: System Reference*, LY28-1011
- *MVS/ESA Integrated Catalog Administration: Access Method Services Reference*, SC26-4500
- *MVS/ESA JCL Reference*, GC28-1829
- *MVS/ESA JCL User's Guide*, GC28-1830
- *MVS/ESA Linkage Editor and Loader User's Guide*, SC26-4510

- *MVS/ESA Message Library: System Messages, Volume 1*, GC28-1812
- *MVS/ESA Message Library: System Messages, Volume 2*, GC28-1813
- *MVS/ESA Message Library: System Codes*, GC28-1815
- *MVS/ESA Planning: Problem Determination and Recovery*, GC28-1629
- *MVS/ESA Service Aids*, GC28-1844
- *MVS/ESA System Generation*, GC28-1825
- *MVS/ESA System Programming Library: Initialization and Tuning*, GC28-1828
- *MVS/ESA VSAM Administration Guide*, SC26-4518
- *MVS/ESA VSAM Catalog Administration: Access Method Services Reference*, SC26-4501

### **MVS/XA (MVS/SP Version 2)**

- *Assembler H Version 2 Application Programming: Guide*, SC26-4036
- *Assembler H Version 2 Application Programming: Language Reference*, GC26-4037
- *MVS/XA Access Method Services Reference Summary for Integrated Catalog Facility*, GX26-3739
- *MVS/XA Catalog Administration Guide*, GC26-4138
- *MVS/XA Configuration Program Guide and Reference*, GC28-1335
- *MVS/XA DADSM and Common VTOC Access Facility Diagnosis Guide*, SY26-3896
- *MVS/XA DADSM Diagnosis Reference*, SY26-3904
- *MVS/XA Data Administration Guide*, GC26-4013
- *MVS/XA Data Administration Guide: Macro Instruction Reference*, GC26-4014
- *MVS/XA Data Administration: Utilities*, GC26-4150
- *MVS/XA Debugging Handbook Volume 1*, LC28-1164
- *MVS/XA Debugging Handbook Volume 2*, LC28-1165
- *MVS/XA Debugging Handbook Volume 3*, LC28-1166
- *MVS/XA Debugging Handbook Volume 4*, LC28-1167
- *MVS/XA Debugging Handbook Volume 5*, LC28-1168
- *MVS/XA Debugging Handbook Volume 6*, LC28-1169
- *MVS/XA Diagnostic Techniques*, LY28-1199

- *MVS/XA ICF Administration: Access Method Services Reference*, GC26-4135
- *MVS/XA JCL User's Guide*, GC28-1351
- *MVS/XA JCL Reference*, GC28-1352
- *MVS/XA Linkage Editor and Loader User's Guide*, GC26-4143
- *MVS/XA Message Library: System Messages, Volume 1*, GC28-1376
- *MVS/XA Message Library: System Messages, Volume 2*, GC28-1377
- *MVS/XA Service Aids*, GC28-1159
- *MVS/XA SPL: Initialization and Tuning*, GC28-1149
- *MVS/XA System Codes*, GC28-1157
- *MVS/XA System Data Administration*, GC26-4010
- *MVS/XA System Generation*, GC26-4148
- *MVS/XA System Macros and Facilities, Volume 1*, GC28-1150
- *MVS/XA VSAM Administration Guide*, GC26-4015
- *MVS/XA VSAM Administration: Macro Instruction Reference*, GC26-4152
- *MVS/XA VSAM Catalog Administration: Access Method Services Reference*, GC26-4136
- *OS/VS2 MVS System Programming Library: Initialization and Tuning Guide*, GC28-0681
- *System 370-XA Principles of Operation*, SA22-7085

### **MVS/370 (MVS/SP Version 1)**

- *IBM System/370 Principles of Operation*, GA22-7000
- *MVS/370 Access Method Services Reference Summary for the Integrated Catalog Facility*, GX26-3745
- *MVS/370 Catalog Administration Guide*, GC26-4053
- *MVS/370 DADSM and Common VTOC Access Facility Diagnosis Guide*, LY27-9509
- *MVS/370 DADSM Diagnosis Reference*, LY27-9510
- *MVS/370 Data Facilities Product Data Administration Guide*, GC26-4058
- *MVS/370 Data Facilities Product Data Administration Macro Instruction Reference*, GC26-4057
- *MVS/370 Data Facilities Product System Data Administration*, GC26-4056
- *MVS/370 Data Facilities Product Utilities Data Administration*, GC26-4065
- *MVS/370 ICF Catalog Administration: Access Method Services Reference*, GC26-4153



- *MVS/370 JCL User's Guide*, GC28-1349
- *MVS/370 JCL Reference*, GC28-1350
- *MVS/370 Linkage Editor and Loader*, GC26-4061
- *MVS/370 Message Library: System Messages, Volume 1*, GC28-1374
- *MVS/370 Message Library: System Messages, Volume 2*, GC28-1375
- *MVS/370 VSAM Catalog Administration: Access Method Services Reference*, GC26-4154
- *MVS/370 VSAM Reference*, GC26-4074
- *MVS/370 VSAM User's Guide*, GC26-4066
- *MVS System Product MVS Diagnostic Techniques*, SY28-1133
- *OS/VS, DOS/VSE and VM/370 Assembler Language*, GC33-4010
- *OS/VS2 Message Library: VS2 System Codes*, GC38-1008
- *OS/VS2 MVS System Programming Library: Service Aids*, GC28-0674
- *OS/VS2 System Programming Library: Debugging Handbook Volume 1*, GC28-1047
- *OS/VS2 System Programming Library: Debugging Handbook Volume 2*, GC28-1048
- *OS/VS2 System Programming Library: Debugging Handbook Volume 3*, GC28-1049



# Index

## Special Characters

### ++APAR MCS

- examples 520
- operands
  - FILES 519
  - REWORK 519
  - RFDSNPFX 520
  - SYSMOD ID 520
- overview 519
- syntax 519

### ++ASMIN

- ASSEM entry 359, 640

### ++ASSIGN MCS

- coding considerations 521
- examples 522
- operands
  - SOURCEID 521
  - TO 521
- overview 521
- syntax 521

### ++BOOK MCS

- See* data element entry
- See* data element MCS
- See* data elements

### ++BSIND MCS

- See* data element entry
- See* data element MCS
- See* data elements

### ++CGM MCS

- See* data element entry
- See* data element MCS
- See* data elements

### ++CLIST MCS

- See* data element entry
- See* data element MCS
- See* data elements

### ++DATA MCS

- See* data element entry
- See* data element MCS
- See* data elements

### ++DELETE MCS

- APPLY processing 86
- coding considerations 531
- examples 531
- operands
  - ALIAS 529
  - name 530
  - SYSLIB 530
- overview 529
- report 479

### ++DELETE MCS *(continued)*

- syntax 529

### ++ENDASMIN

- ASSEM entry 359, 640

### ++ENDLMODIN

- LMOD entry 366, 702

### ++EXEC MCS

- See* data element entry
- See* data element MCS
- See* data elements

### ++FONT MCS

- See* data element entry
- See* data element MCS
- See* data elements

### ++FUNCTION MCS

- coding considerations 534
- examples 534
- operands

FESN 533

FILES 533

REWORK 533

RFDSNPFX 534

SYSMOD ID 534

overview 533

syntax 533

### ++GDF MCS

- See* data element entry
- See* data element MCS
- See* data elements

### ++HELP MCS

- See* data element entry
- See* data element MCS
- See* data elements

### ++HFS MCS

- coding considerations 539
- example 539
- operands

BINARY 535

DELETE 536

DISTLIB 536

LINK 536

name 537

PARM 537

RELFILE 538

RMID 538

SYSLIB 538

TEXT 538

TXLIB 538

VERSION 539

overview 535

syntax 535

## ++HOLD MCS

*See also* exception SYSMODs

*See also* HOLDDATA entry

ACCEPT processing 32

APPLY processing 81

coding considerations 543

examples 543

operands

CLASS 540

COMMENT 541

DATE 541

ERROR 541

FMID 541

REASON 542

SYSMOD ID 543

SYSTEM 541

USER 541

overview 540

report 485

syntax 540

## ++IF MCS

ACCEPT processing 32

APPLY processing 80

coding considerations 546

examples 547

operands

FMID 546

REQ 546

THEN 546

overview 546

syntax 546

## ++IMG MCS

*See* data element entry

*See* data element MCS

*See* data elements

## ++JCLIN MCS

ACCEPT processing 36

APPLY processing 85

coding considerations 549

examples 549

operands

ASM 548

CALLLIBS 548

COPY 548

LKED 548

OPCODE 549

RELFILE 549

TXLIB 549

UPDATE 549

overview 548

reports 471, 473

syntax 548

## ++LMODIN

LMOD entry 366, 702

## ++MAC MCS

ACCEPT processing 42

APPLY processing 92

coding considerations 556

example 556

operands

ASSEM 553

DELETE 554

DISTLIB 554

DISTMOD 554

DISTSRC 554

MALIAS 554

name 555

PREFIX 555

RELFILE 555

RMID 555

SSI 555

SYSLIB 555

TXLIB 556

UMID 556

VERSION 556

overview 553

syntax 553

## ++MACUPD MCS

ACCEPT processing 42

APPLY processing 93

coding considerations 560

example 562

operands

ASSEM 559

DISTLIB 559

DISTMOD 560

DISTSRC 560

MALIAS 560

name 560

PREFIX 560

SYSLIB 560

overview 559

syntax 559

## ++MOD MCS

ACCEPT processing 45

APPLY processing 97

coding considerations 567

examples 567

operands

CSECT 563

DALIAS 564

DELETE 564

DISTLIB 564

LEPARM 565

LKLIB 565

LMOD 565

name 566

RELFILE 566

RMID 566

TALIAS 566

- ++MOD MCS** *(continued)*
  - operands *(continued)*
    - TXLIB 566
    - UMID 567
    - VERSION 567
  - overview 563
  - syntax 563
- ++MOVE MCS**
  - ACCEPT processing 37
  - APPLY processing 86
  - coding considerations 571
  - examples 572
  - operands
    - DISTLIB 570
    - FMID 570
    - LMOD 570
    - MAC 570
    - MOD 570
    - name 570
    - SRC 570
    - SYSLIB 571
    - TODISTLIB 571
    - TOSYSLIB 571
  - overview 570
  - report 479
  - syntax 570
- ++MSG MCS**
  - See data element entry*
  - See data element MCS*
  - See data elements*
- ++NULL MCS**
  - overview 573
  - syntax 573
- ++PARM MCS**
  - See data element entry*
  - See data element MCS*
  - See data elements*
- ++PNL MCS**
  - See data element entry*
  - See data element MCS*
  - See data elements*
- ++PROBJ MCS**
  - See data element entry*
  - See data element MCS*
  - See data elements*
- ++PROC MCS**
  - See data element entry*
  - See data element MCS*
  - See data elements*
- ++PRSRC MCS**
  - See data element entry*
  - See data element MCS*
  - See data elements*
- ++PSEG MCS**
  - See data element entry*
- ++PSEG MCS** *(continued)*
  - See data element MCS*
  - See data elements*
- ++PTF MCS**
  - See also* PTF
  - coding considerations 575
  - examples 575
  - operands
    - FILES 574
    - REWORK 574
    - RFDSNPFX 574
    - SYSMOD ID 575
  - overview 574
  - syntax 574
- ++PUBLB MCS**
  - See data element entry*
  - See data element MCS*
  - See data elements*
- ++RELEASE MCS**
  - coding considerations 578
  - examples 578
  - operands
    - DATE 577
    - ERROR 576
    - FMID 577
    - REASON 577
    - SYSMOD ID 578
    - SYSTEM 576
    - USER 576
  - overview 576
  - report 485
  - syntax 576
- ++RENAME MCS**
  - APPLY processing 86
  - coding considerations 580
  - examples 581
  - operands
    - oldname 580
    - TONAME 580
  - overview 580
  - report 479
  - syntax 580
- ++SAMP MCS**
  - See data element entry*
  - See data element MCS*
  - See data elements*
- ++SKL MCS**
  - See data element entry*
  - See data element MCS*
  - See data elements*
- ++SRC MCS**
  - ACCEPT processing 43
  - APPLY processing 94
  - coding considerations 584
  - examples 584

**++SRC MCS** *(continued)*

operands  
DELETE 582  
DISTLIB 582  
DISTMOD 583  
name 583  
RELFILE 583  
RMID 583  
SSI 583  
SYSLIB 583  
TXLIB 583  
UMID 584  
VERSION 584

overview 582

syntax 582

**++SRCUPD MCS**

ACCEPT processing 43  
APPLY processing 94  
coding considerations 586  
examples 587  
operands

DISTLIB 586  
DISTMOD 586  
name 586  
SYSLIB 586

overview 586

syntax 586

**++TBL MCS**

*See data element entry*

*See data element MCS*

*See data elements*

**++TEXT MCS**

*See data element entry*

*See data element MCS*

*See data elements*

**++USER MCS**

*See data element entry*

*See data element MCS*

*See data elements*

**++USERMOD MCS**

coding considerations 590

examples 590

operands

FILES 589  
REWORK 589  
RFDSNPF 590  
SYSMOD ID 590

overview 589

syntax 589

**++UTIN MCS**

*See data element entry*

*See data element MCS*

*See data elements*

**++UTOUT MCS**

*See data element entry*

**++UTOUT MCS** *(continued)*

*See data element MCS*

*See data elements*

**++VER MCS**

coding considerations 593

examples 594

operands

DELETE 591

FMID 591

NPRES 592

PRE 592

REQ 592

SREL 593

SUP 593

VERSION 593

overview 591

syntax 591

**++ZAP MCS**

ACCEPT processing 46

APPLY processing 100

coding considerations 599

examples 600

operands

DALIAS 598

DISTLIB 598

name 598

TALIAS 598

overview 598

syntax 598

**A****AC=1**

LMOD entry 366, 699

MOD entry 368, 725

**ACCDATE**

SYSMOD entry

distribution zone 371, 762

**ACCEPT**

*See also* ACCEPT command

SYSMOD entry

distribution zone 371, 759

**ACCEPT command**

CHECK processing 21

command termination 23

CSECT processing 47, 48

data set sharing 50

data sets required 18

element selection 38

ENQ considerations 50

examples 24

FMID updating 47

modes of processing

mass mode 30

select mode 30

**ACCEPT command** *(continued)*

moving elements 37

operands

- APARS 7
- ASSEM 7
- BYPASS 8
- CHECK 10
- COMPRESS 10
- EXCLUDE 11
- EXSRCID 11
- FORFMID 11
- FUNCTIONS 12
- GROUP 12
- GROUPEXTEND 12
- JCLINREPORT 14
- NOJCLIN 14
- NOJCLINREPORT 14
- PTFS 14
- RC 14
- REDO 15
- RETRY 15
- REUSE 15
- SELECT 16
- SOURCEID 16
- USERMODS 17

processing

- ++MAC 42
- ++MACUPD 42
- ++MOD 45
- ++SRC 43
- ++SRCUPD 43
- ++ZAP 46
- COMPRESS 41
- data elements 46
- deleted SYSMODs 35
- element installation 41
- inline JCLIN 36
- modules 46
- summary 28
- SYSMOD selection 28

reaccepting a SYSMOD 31

reports 24

RMID updating 47

storing CIFREQ data 50

summary 5

syntax 6

syntax notes 17

SYSMODs

- applicability 31
- installation 34
- processing order 34
- reaccepting 23
- selection 29
- termination 21

UMID updating 47

**ACCEPT command** *(continued)*

updating

- alias names 47
- FMID 47
- RMID 47
- SMPMTS 48
- SMPSCDS 48
- SMPSTS 48
- UMID 48

zone for SET BOUNDARY 5

**ACCEPTCHECK**

- BYPASS operand
- REJECT command 257
- REJECT processing 269

**Access Method Services (AMS)**

*See* AMS utility

**ACCID**

- ACCEPT processing 50
- SYSMOD entry

  - global zone 374, 774

- ZONERENAME processing 436

**ACCJCLIN**

- DLIBZONE entry 363, 675
- JCLIN processing 168

**ACCTIME**

- SYSMOD entry

  - distribution zone 371, 762

**ACDS**

- CONVERT command operand 115
- converting 113

**ACRQ**

- CONVERT command operand 115
- converting 113

**ACTION reason ID 9, 57, 542, 577, 807**

**ADD UCL statement 356**

**adding**

- information to SMPLOG 235
- UCL statements 357

**ALIAS**

*See also* alias names

- ++DELETE MCS operand 529
- data element entry 360, 648
- data element MCS operand 526

**alias names**

*See also* ALIAS

*See also* DALIAS

*See also* MALIAS

*See also* TALIAS

- ++DELETE MCS operand 529
- ACCEPT processing 20, 47
- APPLY processing 70, 102
- load modules 566

**ALIAS statement**

- copy step

  - JCLIN processing 186, 187, 188

**ALIAS statement (continued)**

link-edit step  
JCLIN processing 190

**ALIGN2**

LMOD entry 366, 699  
MOD entry 368, 725

**allocating data sets**

*See also* DDDEF entry  
DDDEF entry 654, 823  
dynamic allocation 823  
module GIMMPDFT 824  
summary 607

**ALLZONES**

LIST command operand 215

**AMODE=24**

LMOD entry 366, 699  
MOD entry 368, 725

**AMODE=31**

LMOD entry 366, 700  
MOD entry 368, 726

**AMODE=ANY**

LMOD entry 366, 700  
MOD entry 368, 726

**AMODE=MIN**

LMOD entry 700  
MOD entry 726

**AMS utility**

default values 741, 787, 836  
EXPORT after conversion 122  
IMPORT after conversion 122  
merging CSIs 423  
OPTIONS entry 369, 741  
UTILITY entry for 741

**AO reason ID 9, 57, 542, 577, 807****APAR fixes**

*See also* ++APAR MCS  
*See also* APARS  
ACCEPT processing 29  
APPLY processing 78  
defining 519  
MCS statement for 519  
naming conventions 810  
REJECT processing 268, 269  
SYSMOD entry for  
distribution zone 371, 759  
global zone 774  
target zone 371, 759

**APARS**

*See also* ++APAR MCS  
*See also* APAR fixes  
ACCEPT command operand 7  
APPLY command operand 55  
LIST command operand 216  
REJECT command operand 257  
UNLOAD command operand 380

**APPDATE**

SYSMOD entry  
target zone 371, 762

**APPID**

APPLY processing 106  
RESTORE processing 347  
SYSMOD entry  
global zone 374, 775  
ZONERENAME processing 436

**APPLY**

*See also* APPLY command  
SYSMOD entry  
target zone 371, 759

**APPLY command**

CHECK processing 71  
command termination 72  
cross-zone processing 83, 105  
CSECT processing 103  
data set sharing 106  
data sets required 67  
deleting load modules 86  
determining SYSLIB 67  
element selection 87  
ENQ considerations 106  
LEPARM processing 568  
modes of processing  
mass mode 78  
select mode 78  
moving elements 86  
moving load modules 86  
operands

APARS 55  
ASSEM 55  
BYPASS 56  
CHECK 58  
COMPRESS 58  
EXCLUDE 58  
EXSRCID 59  
FORFMID 59  
FUNCTIONS 60  
GROUP 60  
GROUPEXTEND 60  
JCLINREPORT 62  
NOJCLIN 62  
NOJCLINREPORT 62  
NUCID 62  
PTFS 62  
RC 62  
REDO 63  
RETRY 63  
REUSE 64  
SELECT 64  
SOURCEID 64  
USERMODS 65

processing  
++MAC 92



**APPLY command** *(continued)*processing *(continued)*

++MACUPD 93

++MOD 97

++SRC 94

++SRCUPD 94

++ZAP 100

compress 91

data elements 101

DELETE on element statements 91

deleted SYSMODs 83

element installation 91

HFS elements 101

inline JCLIN 85

load modules 97

reapplying a SYSMOD 73, 80

renaming load modules 86

storing CIFREQ data 104

summary 53

syntax 54

syntax notes 65

SYSLIB processing 103

SYSMODs

applicability 79

installation 82

operands used for selection 77

processing order 83

selection 78

termination 71

updating

alias names 102

FMID 102

RMID 102

SMPSCDS 103

UMID 103

zone for SET BOUNDARY 53

**APPLYCHECK**

BYPASS operand

ACCEPT command 8

REJECT command 257

REJECT processing 269

**APPTIME**

SYSMOD entry

target zone 371, 762

**ASM***See also* assembler utility

++JCLIN MCS operand 548

JCLIN command operand 168

JCLIN processing 180, 183

OPTIONS entry 369, 741

UTILITY entry for 787

**ASSEM***See also* ASSEM entry

++MAC MCS operand 553

++MACUPD MCS operand 559

**ASSEM** *(continued)*

ACCEPT command operand 7

ACCEPT processing 20, 44

APPLY command operand 55

APPLY processing 69, 95, 96

CONVERT command operand 115

LIST command operand 216

SYSMOD entry

distribution zone 371, 759

target zone 371, 759

UNLOAD command operand 380

**ASSEM entry**

created by JCLIN 183

listing 216, 641

subentries

ASSEMBLER INPUT 640

LASTUPD 640

LASTUPDTYPE 641

summary 640

UCLIN for 359, 643

unloading 380, 642

**ASSEMBLE**

ACCEPT processing 44, 45

APPLY processing 95, 96

MOD entry 368, 723

**ASSEMBLER INPUT**

ASSEM entry 640

**assembler utility**

++JCLIN MCS operand 548

ACCEPT processing 44

APPLY processing 95, 96

default values 741, 787, 789, 836

GENERATE processing 157, 162

JCLIN processing 168, 183

OPTIONS entry 369, 741

specifying on JCLIN 168, 548

UTILITY entry for 741, 787

**assemblies***See also* ASSEM entry

macros causing

ACCEPT processing 44

APPLY processing 96

reusing

ACCEPT processing 45

APPLY processing 97

source

ACCEPT processing 44

APPLY processing 95

**ASSIGN***See* ++ASSIGN MCS**assigning source IDs to SYSMODs**

++ASSIGN MCS

RECEIVE command

++ASSIGN processing 253

SOURCEID operand 242

**authorized installation-wide exit routines 798**

*See also* installation-wide exit routines for SMP/E

**automatic call libraries**

contrasted with LINK command 201

LIBRARY statement to exclude modules from automatic library search 195

SYSLIB DD statement in link-edit steps 196

**automatic reaccept 23****automatic reapply 73****automatic rereceive 252**

## B

**backing off changes in the target libraries**

(RESTORE command) 333

**BACKUP**

*See also* BACKUP entries

LIST command operand 216

**BACKUP entries**

ACCEPT processing 48

APPLY processing 85, 86, 91, 103

listing 216, 645

RESTORE processing 338, 342, 343, 346

subentries 645

summary 645

UCLIN for 359, 646

**batch local shared resources (LSR), performance improved by 867****BDY**

*See* BOUNDARY

**BEGINDATE**

REPORT ERRSYSMODS command operand 299

**BINARY**

++HFS MCS operand 535

HFS entry 365, 687

**binder**

*See also* link-edit utility

default values 836

UTILITY entry for 161, 789

**BLOCK**

DDDEF entry 361, 654

**BLSR**

*See* batch local shared resources (LSR), performance improved by

**BOOK member for online manuals**

MCS for

*See* data element MCS

**bookshelf**

MCS for

*See* data element MCS

**BOUNDARY**

SET command operand 349

**BUFND parameter 867****BUFNI parameter 867****building load modules 90****BYPASS**

ACCEPT command operand 8

ACCEPT processing 22, 39

APPLY command operand 56

APPLY processing 72, 88

LIST command operand 216

RECEIVE command operand 239

RECEIVE processing 251

REJECT command operand 257

REJECT processing 269

RESTORE command operand 333

RESTORE processing 337

SYSMOD entry

distribution zone 371, 760

target zone 371, 760

## C

**CALL**

LMOD entry 700

passed based on existence of CALLLIBS

subentry 98, 787, 789

**callable services**

including modules from another product 196

**calling SMP/E 813****CALLLIBS**

++JCLIN MCS operand 168

APPLY processing 70, 98, 99

GENERATE processing 158

LINK processing 207

LMOD entry 698

REPORT CALLLIBS command operand 275

RESTORE processing 345

restriction on use with previous releases of

SMP/E 196, 698

restrictions 865

specifying on JCLIN 548

**CALLLIBS Summary report 442****CASE**

LMOD entry 700

**CATALOG**

DDDEF entry 361, 655

**cataloged procedure for SMP/E 813, 815****Causar SYSMOD Summary report 444****CCC value of FMID 810****CDS**

CONVERT command operand 115

converting 113

**CHANGE**

ZONEEDIT command operand 407

**CHANGE statement**

JCLIN processing 193

RESTORE processing restriction 342

**CHECK**

ACCEPT command operand 10  
 ACCEPT processing 21, 41  
 APPLY command operand 58  
 APPLY processing 71, 90  
 RESTORE command operand 333

**checking data set entries 211****checking done by the REPORT CALLLIBS command**

load modules with CALLLIBS SUBENTRY list 275

**checking done by the REPORT CROSSZONE****command**

conditional requisites 287  
 cross-zone requisites 287

**checking done by the REPORT ERRSYSMODS****command**

exception SYSMODs 299  
 PE-PTFs 299

**checking done by the REPORT SOURCEID****command**

SOURCEIDs 309

**checking done by the REPORT SYSMODS command**

missing SYSMODs 317

**CIDTABL**

restrictions 865

**CIFREQ**

*See also* ++IF MCS

*See also* conditional requisites

ACCEPT processing 32, 50

APPLY processing 80, 104

SYSMOD entry

distribution zone 372, 373, 760

target zone 372, 373, 760

**CLASS**

++HOLD MCS operand 540

ACCEPT processing 33, 34

APPLY processing 81, 82

naming conventions 808

values

ERREL 8, 56, 541, 808

UCLIN 8, 56, 541, 808

**cleaning up data sets 107****CLEANUP command**

data set sharing 111

ENQ considerations 111

examples 109

operands

COMPRESS 108

RC 108

processing 110

reports 446

summary 107

summary report 446

syntax 107

zone for SET BOUNDARY 107

**CLIST**

MCS for

*See* data element MCS

**command phases 832****command syntax rules 2****COMMENT**

++HOLD MCS operand 541

**COMP**

*See also* compress utility

OPTIONS entry 369, 741

UTILITY entry 787

**COMPAREDTO**

REPORT SYSMODS command operand 317

**comparing two zones**

LIST command 211

REPORT CROSSZONE command 287

**compatibility with previous releases of SMP/E 865****COMPRESS**

ACCEPT command operand 10

ACCEPT processing 41

APPLY command operand 58

APPLY processing 91

CLEANUP command operand 108

CLEANUP processing 110

REJECT command operand 258

RESTORE command operand 334

RESTORE processing 343

**compress utility**

ACCEPT processing 41

APPLY processing 58, 91

CLEANUP processing 110

CONVERT processing 122

default values 741, 787, 836

OPTIONS entry 369, 741

REJECT processing 258

RESTORE processing 334, 343

UTILITY entry for 741, 787

**compressing data sets**

ACCEPT processing 41

APPLY processing 58, 91

CLEANUP processing 110

CONVERT processing 122

REJECT processing 258

RESTORE processing 334, 343

**CONCAT**

DDDEF entry 361, 655

**concatenating data sets**

DDDEF entry 655

not allowed for SMPPTS 615

not allowed in JCLIN 181

SYSLIB data sets

++MAC 556

++MACUPD 561

order 622

SMPPTS 613

use 622

## conditional requisites

*See also* ++IF MCS

ACCEPT processing 32, 50

APPLY processing 80, 104

checking for across zones

REPORT CROSSZONE command 287

defining 546

## Consolidated Software Inventory

*See* CSI

## CONTENT

entries defined 430

ZONEMERGE command operand 424

## continuation character for link-edit statements 182

## CONVERT command

compressing CSI after conversion 122

converting CSI data sets 119

converting SCDS data sets 121

cross-reference of data sets and entries 118

data set sharing 141

data sets required 119

ENQ considerations 141

examples 123

FMID entry 140

operands

ACDS 115

ACRQ 115

ASSEM 115

CDS 115

CRQ 116

DLIB 116

LMOD 116

MAC 116

MCS 116

MOD 116

OLDCSI 116

PTS 117

RC 117

SCDS 117

SRC 118

SYS 118

SYSMOD 118

other SMP4 data sets

general case 138

processing

PTS SYSTEM entry 140

special cases 139

recovery 123

summary 113

syntax 114

syntax notes 118

SYSTEM entry 140

usage notes 119

zone for SET BOUNDARY 114

## converting from SMP4 113

## COPY

*See also* copy utility

++JCLIN MCS operand 548

JCLIN command operand 168

JCLIN processing 180, 185

LMOD entry 366, 698

OPTIONS entry 369, 742

UTILITY entry for 787

## copy utility

ACCEPT processing 42, 47

APPLY processing 93, 101

default values 742, 787, 836

GENERATE processing 164

JCLIN processing 185

OPTIONS entry 369, 742

specifying on JCLIN 168, 548

UTILITY entry for 742, 787

## copying a CSI data set to recover unused space after conversion 122

## copying a zone

from a sequential data set

ZONEIMPORT command 417

into a different CSI data set

ZONECOPY command 391

into a sequential data set

ZONEEXPORT command 413

within the same CSI data set

ZONEMERGE command 423

## copying data set entries 379

## COPYJOB job, built by GENERATE for copied elements 162

## corequisite SYSMODs

*See also* REQ

++IF MCS 546

ACCEPT processing 32

APPLY processing 80

defining 592

## cover letters

listing 222

printing 222

## creating installation job streams (GENERATE command) 149

## cross-product load modules

example 204

## cross-zone load modules

APPLY processing 83, 87, 105

checking for duplicate modules 91

creating 201

deleting 87

example 204

GENERATE processing 158

JCLIN processing 172, 198

LINK command 201

listing LMOD entries for 228

renaming 87

**cross-zone load modules** *(continued)*

RESTORE processing 346  
 restrictions 866  
 unloading entries for 387

**cross-zone modules**

APPLY processing 83, 105  
 deleting 91  
 GENERATE processing 158  
 LINK command 201  
 listing MOD entries for 228  
 reincluding in load modules with a SYSLIB  
 allocation 100  
 RESTORE processing 346  
 restrictions 866  
 unloading entries for 386

**Cross-Zone Requisite SYSMOD report 447****cross-zone requisites**

checking for  
 REPORT CROSSZONE command 287

**cross-zone subentries**

restriction on using zone with previous releases of  
 SMP/E 201  
 ZONECOPY processing 393, 396  
 ZONEDELETE processing 401, 402  
 ZONEEDIT processing 405  
 ZONEEXPORT processing 415, 416  
 ZONEIMPORT processing 419, 421  
 ZONEMERGE processing 425  
 ZONERENAME processing 439

**Cross-Zone Summary report 449****CROSSZONE**

REPORT CROSSZONE command operand 287

**CRQ**

CONVERT command operand 116  
 converting 113

**CSECT**

++MOD MCS operand 563  
 ACCEPT processing 48  
 APPLY processing 103  
 deleting 103  
 MOD entry 368, 723

**CSI**

ASSEM entry  
 distribution zone 640  
 target zone 640  
 data element entry  
 distribution zone 648  
 target zone 648  
 DDDEF entry  
 distribution zone 654  
 global zone 654  
 target zone 654  
 defining 609  
 DLIB entry  
 distribution zone 670  
 target zone 670

**CSI** *(continued)*

DLIBZONE entry 675  
 FMIDSET entry 679  
 GLOBALZONE entry 682  
 HOLDDATA entry 695  
 LMOD entry  
 distribution zone 698  
 target zone 698  
 merging  
 AMS REPRO command 423  
 ZONEMERGE command 423  
 parameter  
 EXEC statement for GIMSMP 813  
 processing 137  
 sharing 834

**CYLINDERS**

DDDEF entry 361, 654

**D****DA**

*See* DATASET

**DALIAS**

++MOD MCS operand 564  
 ++ZAP MCS operand 598  
 ACCEPT processing 20  
 APPLY processing 70  
 MOD entry 368, 724  
 RECEIVE processing 251

**data element entry**

listing 649  
 subentries 648  
 summary 648  
 UCLIN for 360, 652  
 unloading 651

**data element MCS**

coding considerations 528  
 examples 528  
 operands  
 ALIAS 526  
 DELETE 526  
 DISTLIB 527  
 name 527  
 RELFILE 527  
 RMID 527  
 SYSLIB 527  
 TXLIB 527  
 VERSION 527  
 overview 523  
 syntax 526

**data elements**

*See also* data element entry  
*See also* data element MCS  
 ACCEPT processing 46  
 adding 523

**data elements** *(continued)*

- APPLY processing 101
- defining 523
- deleting
  - ACCEPT processing 35
  - APPLY processing 83, 91
  - MCS operand 526
- listing 217, 649
- replacing
  - ACCEPT processing 46
  - APPLY processing 101
  - MCS operand 523
- SYSMOD entry
  - distribution zone 760
  - global zone 775
  - target zone 760
- unloading 381, 651

**data set organization** 627**data sets**

- See also* DDDEF entry
- defining 607
- distribution library 607
- dynamically allocating 823
- INFILE 607
- link library 608
- LKLIB 608
- OUTFILE 608
- PARMLIB 608
- SMP\_CNTL 609
- SMP\_CSI 609
- SMP\_DEBUG 610
- SMP\_HOLD 610
- SMP\_JCLIN 610
- SMP\_LIST 611
- SMP\_LOG 611
- SMP\_LOGA 612
- SMP\_LTS 612
- SMP\_MTS 613
- SMP\_nnnnn 621
- SMP\_OBJ 614
- SMP\_OUT 614
- SMP\_PTFIN 614
- SMP\_PTS 615
- SMP\_PUNCH 615
- SMP\_RPT 616
- SMP\_SCDS 616
- SMP\_SNAP 617
- SMP\_STS 617
- SMP\_TLIB 618
- SMP\_WRK1 619
- SMP\_WRK2 620
- SMP\_WRK3 620
- SMP\_WRK4 621
- SMP\_WRK6 621
- SYSLIB 622

**data sets** *(continued)*

- SYSPRINT 622
  - SYSPUNCH 623
  - SYSUT1 623
  - SYSUT4 624
  - target library 624
  - text library 624
  - TXLIB 624
  - zonename 624
- data transformation service routine (GIMDTS)** 803
- DATACLAS**
  - DDDEF entry 655
- DATASET**
  - DDDEF entry 361, 656
- DATE**
  - ++HOLD MCS operand 541
  - ++RELEASE MCS operand 577
- DATE parameter, EXEC statement for GIMSMP** 813
- DC**
  - LMOD entry 366, 700
  - MOD entry 368, 726
- DDDEF**
  - See* DDDEF entry
- DDDEF entry**
  - instead of DD statements in cataloged procedure 815
  - LIST command operand 216
  - listing 216, 660
  - subentries
    - BLOCK 654
    - CATALOG 655
    - CONCAT 655
    - CYLINDERS 654
    - DATACLAS 655
    - DATASET 656
    - DELETE 655
    - DIR 656
    - DSNTYPE 656
    - DSPREFIX 657
    - KEEP 655
    - MGMTCLAS 657
    - MOD 657
    - NEW 657
    - OLD 657
    - PATH 658
    - PROTECT 658
    - SHR 657
    - SPACE 659
    - STORCLAS 659
    - SYSOUT 659
    - TRACKS 654
    - UNIT 660
    - VOLUME 660
    - WAIT 660
  - summary 654

**DDDEF entry (continued)**

target zone  
 GENERATE processing 161  
 UCLIN for  
 distribution zone 361, 663  
 global zone 663  
 target zone 361, 663  
 UNLOAD command operand 381  
 unloading 381, 662  
 updating multiple entries  
 ZONEEDIT command 405  
 used for dynamic allocation 823

**ddnames**

HFS copy utility  
 alternate values used for APPLY processing 102

**DEBUG command**

data sets required 145  
 examples 145  
 operands  
 DUMPMSG 144  
 DUMPOFF 144  
 DUMPON 144  
 DUMPRPL 144  
 MSGMODID 145  
 SNAP 144  
 processing 148  
 summary 143  
 syntax 143  
 zone for SET BOUNDARY 143

**debugging SMP/E problems 143****default utilities used by SMP/E 787****defaults**

OPTIONS entry  
 distribution zone 675  
 global zone 682  
 on the SET command 741  
 target zone 782

SMPOUT 827

**SMPTLIB**

data set space 742  
 disposition 619

**subentries**

NOPURGE 744  
 NOREJECT 744  
 PAGELEN 744  
 PEMAX 744  
 RETRYDDN 745  
 SAVEMTS 745  
 SAVESTS 745

SYSPRINT 827

**utility programs**

access method services 741, 787, 836  
 assembler 741, 787, 789, 836  
 binder link-edit utility 836  
 compress 741, 787, 836  
 copy 742, 787, 836

**defaults (continued)****utility programs (continued)**

HFS copy utility 836  
 link-edit utility 743, 787, 789, 836  
 retry 744, 787, 836  
 superzap 746, 787, 836  
 update 745, 787, 836

**defining data sets**

See also DDDEF entry  
 data set descriptions 607  
 DDDEF entry 654, 823  
 dynamic allocation 823  
 module GIMMPDFT 824

**DEFINITION**

entries defined 430  
 ZONEMERGE command operand 424

**DEL UCL statement 356****DELBY**

SYSMOD entry  
 distribution zone 373, 760  
 target zone 373, 760

**DELETE**

See also ++DELETE MCS  
 ++HFS MCS operand 536  
 ++MAC MCS operand 554  
 ++MOD MCS operand 564  
 ++SRC MCS operand 582  
 ++VER MCS operand 591  
 ACCEPT processing 12, 16, 35  
 APPLY processing 59, 60, 64, 83  
 APPLY processing 91  
 data element MCS operand 526  
 DDDEF entry 361, 655  
 LIST command operand 216  
 MCS statement 529  
 RESTORE processing 343  
 SYSMOD entry  
 distribution zone 371, 760  
 global zone 775  
 target zone 371, 760  
 UNLOAD command operand 381

**DELETE reason ID 9, 57, 542, 577, 807****deleted elements, restoring 343****Deleted SYSMOD report 453****deleted SYSMODs**

ACCEPT processing 49  
 APPLY processing 104  
 dummy entry for 50, 104

**DELETFMID**

REJECT command operand 258  
 REJECT processing 271

**deleting changes from target libraries (RESTORE command) 333****deleting data set entries**

CLEANUP command 107

## deleting data set entries *(continued)*

- REJECT command 255
- UCL statements 357
- ZONEDELETE command 399

## deleting elements

- data elements 35, 83, 526
- HFS elements 536
- macros 35, 83, 554
- modules 35, 83, 564
- source 35, 83, 582

## deleting functions

- ++VER MCS 591
- ACCEPT processing 35
- APPLY processing 83
- dummy function SYSMOD 596
- example of 596

## deleting load modules

- ++DELETE MCS 529
- APPLY processing 84, 86

## deleting zones 399

### DELLMOD

- SYSMOD entry
  - distribution zone 371, 760
  - target zone 371, 760

## DEP reason ID 9, 57, 542, 577, 807

## DEQ command

- GRS considerations 833
- used for zone sharing 833

## diagnosing SMP/E problems 143

## dialog HELP panels

- MCS for
  - See data element MCS

## dialog messages

- MCS for
  - See data element MCS

## dialog panels

- MCS for
  - See data element MCS

## dialog skeletons

- MCS for
  - See data element MCS

## dialog tables

- MCS for
  - See data element MCS

## dialogs, packaging

- MCS for
  - See data element MCS

## DIR

- DDDEF entry 361, 656

## directory information for pathname 658

## displaying data set entries 211

## DISTLIB

- See also distribution libraries
- ++HFS MCS operand 536
- ++MAC MCS operand 554

## DISTLIB *(continued)*

- ++MACUPD MCS operand 559
- ++MOD MCS operand 564
- ++MOVE MCS operand 570
- ++SRC MCS operand 582
- ++SRCUPD MCS operand 586
- ++ZAP MCS operand 598
- ACCEPT processing 19
- APPLY processing 68
- data element entry 360, 648
- data element MCS operand 527
- HFS entry 365, 687
- MAC entry 367, 715
- MOD entry 368, 724
- SRC entry 370, 750

## DISTMOD

- ++MAC MCS operand 554
- ++MACUPD MCS operand 560
- ++SRC MCS operand 583
- ++SRCUPD MCS operand 586
- ACCEPT processing 20
- APPLY processing 69

## distribution libraries

- compressing 41
- defining 607
- installing SYSMODs in 5, 53

## distribution library

- See DISTLIB
- See DLIB entry

## distribution zone

- ASSEM entry 640
- data element entry 648
- DDDEF entry 654
- DLIB entry 670
- DLIBZONE entry 675
- HFS entry 687
- LMOD entry 698
- MAC entry 715
- MOD entry 723
- sharing 833
- SRC entry 750
- SYSMOD entry 759
- updating with JCLIN data 36
- zone description provided by user 676

## DISTSRC

- ++MAC MCS operand 554
- ++MACUPD MCS operand 560
- ACCEPT processing 20
- APPLY processing 69

## DLIB

- See also DISTLIB
- See also distribution libraries
- See also DLIB entry
- CONVERT command operand 116
- LIST command operand 217



**DLIB (continued)**

UNLOAD command operand 381

**DLIB entry**

created by JCLIN 186

listing 217, 671

subentries

LASTUPD 671

LASTUPDTYPE 671

SYSLIB 671

UCL syntax 671

summary 670

UCLIN for 363, 673

unloading 381, 672

used to determine SYSLIB 186

**DLIBZONE***See also* DLIBZONE entry

LIST command operand 217

REPORT CROSSZONE command operand 287

ZONEDELETE command operand 399

**DLIBZONE entry**

listing 217, 676

subentries

ACCJCLIN 675

name 675

OPTIONS 675

RELATED 675

SREL 676

UCL syntax 675

ZDESC 676

summary 675

UCLIN for 363, 677

**DLMOD**

SYSMOD entry

distribution zone 760

target zone 760

**DOC reason ID 9, 57, 542, 577, 807****DSNTYPE**

DDDEF entry 656

**DSPREFIX**

conflict with RFDSNPFX value 742

DDDEF entry (global zone only) 361, 657

OPTIONS entry 369, 742

RECEIVE processing 250

SYSMOD entry

global zone 777

**DSSPACE**

OPTIONS entry 369, 742

RECEIVE processing 250

**dummy data sets**

defined in DDDEF entry 656

not allowed for SMPPTS 615

**dummy entry for deleted SYSMODs 50, 104****dummy entry for superseded SYSMODs 49, 104****dummy function SYSMOD to delete another function 596****dumping data with the DEBUG command**

SMP/E storage and control blocks 143

VSAM RPL control blocks 143

**DUMPMSG**

DEBUG command operand 144

**DUMPOFF**

DEBUG command operand 144

**DUMPON**

DEBUG command operand 144

**DUMPRPL**

DEBUG command operand 144

**dumps 143****dynamic allocation**

CSI parameter on EXEC statement for

GIMSMP 813

DDDEF entry 654, 823

effect of SET command 350, 352

processing 827

SMPTLIB 250

sources of information

DDDEF entries 823

GIMMPDFT 824

standard defaults 827

summary 823

**Dynamic Allocation report***See* File Allocation report**DZONE***See also* DLIBZONE

REPORT CROSSZONE processing 296

**E****E***See* EXCLUDE**EC reason ID 9, 57, 542, 577, 807****element**

LIST command operand 217

UNLOAD command operand 381

**Element Summary report 455****elements**

deleted

APPLY processing 91

RESTORE processing 343

deleting

ACCEPT processing 35

APPLY processing 83

data elements, MCS for 526

HFS elements, MCS for 536

macros, MCS for 554

modules, MCS for 564

source, MCS for 582

moving

++MOVE MCS 570

ACCEPT processing 37

APPLY processing 86

RESTORE processing 345

## ELEMMOV

- SYSMOD entry
  - distribution zone 371, 761
  - target zone 371, 761

## EMOVE

- SYSMOD entry
  - distribution zone 761
  - target zone 761

## END

- EXEC statement parameter for GIMSMP 814, 833

## ENDDATE

- REPORT ERRSYSMODS command operand 300

## ENDUCL command

- syntax 356

## ENQ command

- GRS considerations 833
- used for zone sharing 833

## entries

- ASSEM entry 640
- BACKUP entries 645
- CSI 629
- data element entry 648
- DDDEF entry 654
- DLIB entry 670
- DLIBZONE entry 675
- FMIDSET entry 679
- GLOBALZONE entry 682
- HFS entry 687
- HOLDDATA entry 695
- LMOD entry 698
- MAC entry 715
- MCS entry 721
- MOD entry 723
- MTSMAC entry 739
- OPTIONS entry 741
- relationship between
  - distribution zone 629
  - global zone 629
  - target zone 629
- SMPMTS 739
- SMPPTS 721
- SMPSCDS 645
- SMPSTS 757
- SRC entry 750
- STSSRC entry 757
- SYSMOD entry
  - distribution zone 759
  - global zone 774
  - target zone 759
- TARGETZONE entry 782
- UTILITY entry 787
- ZONESET entry 794

## ENTRY statement

- JCLIN processing 194

## environment

- defining 591

## ERR

- See ERROR

## ERREL class value

- ACCEPT command 8, 56, 541, 808

## ERROR

- See also error reason IDs
- ++HOLD MCS operand 541
- ++RELEASE MCS operand 576
- LIST command operand 217
- reason IDs 542, 577, 807
- SYSMOD entry
  - distribution zone 371, 761
  - global zone 775
  - target zone 371, 761
- UNLOAD command operand 381

## error reason IDs

- ACCEPT processing 33
- APPLY processing 81
- naming conventions 807

## errors, debugging 143

## ERRSYSMODS

- REPORT ERRSYSMODS command operand 300

## exception SYSMOD management

- See also exception SYSMODs
- ++HOLD MCS 540, 576
  - operands 540, 576
- processing
  - ACCEPT 32
  - APPLY 81
  - RECEIVE 253
  - REJECT 272
  - RESTORE 338

## Exception SYSMOD report 460

## exception SYSMODs

- See also exception SYSMOD management
- ACCEPT processing 32
- APPLY processing 81
- checking for
  - REPORT ERRSYSMODS command 299
- holding 540
- RECEIVE processing 253
- REJECT processing 272
- releasing 576
- report 485
- resolving (REPORT ERRSYSMODS command) 299
- RESTORE processing 338

## EXCLUDE

- ACCEPT command operand 11
- ACCEPT processing 29
- APPLY command operand 58
- APPLY processing 78
- RECEIVE command operand 239

**EXCLUDE** (*continued*)

RECEIVE processing 242  
 REJECT command operand 258  
 REJECT processing 268

**EXCLUDEZONE**

REJECT command operand 258  
 REJECT processing 268, 269

**excluding modules from automatic library**

search 195

**EXEC**

MCS for  
 See data element MCS

**EXEC statement**

CSI=dsname 813  
 DATE=date 813  
 LANGUAGE 814  
 PARM 813, 814  
 PARM parameter 833  
 PROCESS=END 814, 833  
 PROCESS=WAIT 814, 833

**exit routines**

See installation-wide exit routines for SMP/E

**EXPAND statement**

++ZAP input 599  
 JCLIN processing 194

**explicitly deleting functions**

ACCEPT processing 35  
 APPLY processing 83

**EXPORT**

See also AMS utility  
 See also ZONEEXPORT command  
 after conversion 122

**EXRF reason ID 9, 57, 542, 577, 808****EXRTYDD**

OPTIONS entry 742  
 restrictions 866

**EXSRCID**

ACCEPT command operand 11  
 APPLY command operand 59  
 LIST command operand 217  
 UNLOAD command operand 381

**external HOLDDATA 487, 540, 695****F****FESN**

++FUNCTION MCS operand 533  
 SYSMOD entry  
 distribution zone 371, 761  
 global zone 775  
 target zone 371, 761

**FETCHOPT**

LMOD entry 700  
 MOD entry 726

**File Allocation report 463****file skeletons for dialogs**

MCS for  
 See data element MCS

**FILES**

++APAR MCS operand 519  
 ++FUNCTION MCS operand 533  
 ++PTF MCS operand 574  
 ++USERMOD MCS operand 589

**FMID**

See also FMID entry  
 ++HOLD MCS operand 541  
 ++IF MCS operand 546  
 ++MOVE MCS operand 570  
 ++RELEASE MCS operand 577  
 ++VER MCS operand 591  
 ACCEPT processing 38, 47  
 APPLY processing 87, 102  
 BYPASS operand  
 RECEIVE command 239  
 data element entry 360, 648  
 FMIDSET entry 364, 679  
 GLOBALZONE entry 364, 682  
 HFS entry 365, 687  
 MAC entry 367, 715  
 MOD entry 368, 724  
 RECEIVE processing 251  
 SRC entry 370, 750  
 SYSMOD entry  
 distribution zone 371, 761  
 global zone 775  
 target zone 371, 761

**FMID entry**

conversion 140

**FMIDSET**

See also FMIDSET entry  
 LIST command operand 218

**FMIDSET entry**

listing 218, 679  
 subentries  
 FMID 679  
 UCL syntax 679  
 summary 679  
 UCLIN for 364, 680

**font (FOCA)**

MCS for  
 See data element MCS

**FORFMID**

ACCEPT command operand 11  
 ACCEPT processing 29  
 APPLY command operand 59  
 APPLY processing 78  
 GENERATE command operand 150  
 GENERATE processing 155, 157, 158, 159  
 LIST command operand 218

**FORFMID** (*continued*)

- RECEIVE command operand 239
- REJECT command operand 259
- REJECT processing 268, 270
- REPORT CROSSZONE command operand 288
- REPORT CROSSZONE processing 296
- REPORT ERRSYSMODS command operand 301
- REPORT ERRSYSMODS processing 305
- UNLOAD command operand 382

**FORZONE**

- REPORT CROSSZONE command operand 288
- REPORT CROSSZONE processing 295, 296

**FROMZONE**

- LINK command operand 202

**FULLGEN reason ID 9, 57, 542, 577, 808****function SYSMODs**

*See also ++FUNCTION MCS*

deleting

- ++VER MCS 591
- ACCEPT processing 35
- APPLY processing 83
- explicit deletion 35, 83
- implicit deletion 35, 83
- naming conventions 810
- reaccepting 40
- reapplying 90

**FUNCTION**

- SYSMOD entry
  - distribution zone 761
  - global zone 775
  - target zone 761

**FUNCTIONS**

- ACCEPT command operand 12
- ACCEPT processing 29
- APPLY processing 78
- defining 533
- REJECT command operand 259
- REJECT processing 268, 269

**FUNCTIONS**

- See also ++FUNCTION MCS*
- APPLY command operand 60
- LIST command operand 219
- MCS statement 533
- UNLOAD command operand 383

**FUNCTION**

- SYSMOD entry
  - distribution zone 371
  - target zone 371

**G****GDF graphics panel**

MCS for

*See data element MCS*

**GENASM**

- MAC entry 367, 716

**GENERATE command**

- cross-zone load modules 158
- cross-zone modules 158
- data set sharing 165
- data sets required 151
- ENQ considerations 165
- examples 152
- operands
  - FORFMID 150
  - JOBCARD 150
  - RC 150
  - REPLACE 150
- processing 155
- SMPPUNCH output 152
- summary 149
- summary report 467
- syntax 149
- usage notes 152
- zone for SET BOUNDARY 149

**generating installation job streams (GENERATE command) 149****GIMDFUT 835****GIMDTS**

- ACCEPT processing 47
- APPLY processing 101
- inline data elements 528, 539
- summary 803

**GIMGNIAP 164, 869****GIMMPDFT**

- GENERATE processing 161
- preparing with GIMZPDFT 824
- used for dynamic allocation 824

**GIMMPUXD 797, 798****GIMMPUXP 797, 798****GIMOPCDE 171, 184****GIMSMP 813, 815****GIMUTTBL 788, 835****GIMZPDFT 824****GIMZPOOL**

- not used for ZONERENAME processing 438
- required before ZONECOPY processing 393
- required before ZONEIMPORT processing 418

**global zone**

- DDDEF entry 654
- FMIDSET entry 679
- GLOBALZONE entry 682
- HOLDDATA entry 695
- OPTIONS entry 741
- sharing 833, 834
- SYSMOD entry 774
- unexpected changes for (pending updates) 834
- UTILITY entry 787
- zone description provided by user 683

**global zone** *(continued)*

ZONASET entry 794

**GLOBALZONE***See* GLOBALZONE entry**GLOBALZONE entry**

LIST command operand 219

listing 219, 684

subentries

FMID 682

OPTIONS 682

SREL 682

UCL syntax 682

ZONEDESCRIPTION 683

ZONEINDEX 683

summary 682

UCLIN for 364, 684

**graphics image for online book**

MCS for

*See* data element MCS**graphics page segment for online book**

MCS for

*See* data element MCS**graphics panel (GDF)**

MCS for

*See* data element MCS**graphics source for online book**

MCS for

*See* data element MCS**GROUP**

ACCEPT command operand 12

ACCEPT processing 30

APPLY command 60

APPLY processing 78

RESTORE command operand 334

RESTORE processing 337, 342

**GROUPEXTEND**

ACCEPT command operand 12

ACCEPT processing 30

APPLY command operand 60

APPLY processing 78

**GRS enqueue names used by SMP/E 833****H****held SYSMODs***See* exception SYSMODs**HELP information and panels for dialogs**

MCS for

*See* data element MCS**HEWLH096 utility***See* link-edit utility**HFS***See also* HFS elements*See also* hierarchical file system (HFS)

LIST command operand 219

**HFS** *(continued)*

MCS statement 535

SYSMOD entry

distribution zone 371, 761

global zone 775

target zone 371, 761

UNLOAD command operand 383

**HFS elements***See also* HFS entry

++HFS MCS 536

adding 535

APPLY processing 101

defining 535

deleting 536

replacing

++HFS 535

APPLY processing 101

**HFS entry***See also* HFS elements

++HFS MCS 535

listing 219, 689

subentries

BINARY 687

DISTLIB 687

FMID 687

LASTUPD 688

LASTUPDTYPE 688

LINK 688

PARM 688

RMID 689

SYSLIB 689

TEXT 687

UCL syntax 687

summary 687

UCLIN for 365, 693

unloading 383, 691

**HFSCOPY***See also* hierarchical file system (HFS)

OPTIONS entry 369, 743

**HFSCOPY subentry**

restrictions 866

**HFSINST job, built by GENERATE for HFS**

elements 163, 869

**hierarchical file system (HFS)**

copy utility

default values 787, 836

interaction with HFS element entry 789

OPTIONS entry 369, 743

parameters used for APPLY processing 102

UTILITY entry for 743

ddnames, alternate values used during APPLY processing 102

load modules residing in

JCLIN for 178

LIBRARYDD comment 196, 197

SYSLIB DD statement in link-edit steps 196

**hierarchical file system (HFS) (continued)**

- load modules residing in (continued)
- SYSLMOD DD statement in link-edit steps 197

**high-level languages**

- including modules from another product 196

**hiperspaces, performance improvements through 867**

**HOLD**

- See ++HOLD MCS
- See HOLD reason IDs

**HOLD reason IDs**

- ACCEPT processing 32
- APPLY processing 81
- class values
  - ++HOLD MCS 540
  - ACCEPT processing 34
  - APPLY processing 82
  - ERREL 8, 56, 541, 808
  - naming conventions 808
  - UCLIN 8, 56, 541, 808
- error reason IDs
  - ++HOLD MCS 542
  - ++RELEASE MCS 577
  - ACCEPT processing 33
  - APPLY processing 81
  - naming conventions 807
- naming conventions 807
- RECEIVE processing 253
- system reason IDs
  - ++HOLD MCS 542
  - ++RELEASE MCS 577
  - ACCEPT processing 33
  - ACTION 9, 57, 542, 577, 807
  - AO 9, 57, 542, 577, 807
  - APPLY processing 82
  - DELETE 9, 57, 542, 577, 807
  - DEP 9, 57, 542, 577, 807
  - DOC 9, 57, 542, 577, 807
  - EC 9, 57, 542, 577, 807
  - EXRF 9, 57, 542, 577, 808
  - FULLGEN 9, 57, 542, 577, 808
  - IOGEN 9, 57, 542, 577, 808
  - MSGSKEL 9, 57, 542, 577, 808
  - MVSCP 9, 57, 543, 578, 808
  - naming conventions 807
- user reason IDs
  - ++HOLD MCS 543
  - ++RELEASE MCS 578
  - ACCEPT processing 34
  - APPLY processing 82
  - naming conventions 808

**HOLDCLASS**

- BYPASS operand
  - ACCEPT command 8
  - ACCEPT processing 33
  - APPLY command 56

**HOLDCLASS (continued)**

- BYPASS operand (continued)
- APPLY processing 81

**HOLDDATA**

- See also ++HOLD MCS
- See also exception SYSMODs
- See also HOLD reason IDs
- See also HOLDDATA entry
- ACCEPT processing 32, 50
- APPLY processing 81
- external 540, 695
- internal 540, 578, 695
- LIST command operand 219
- purged at RESTORE 347
- RECEIVE command operand 240
- RECEIVE processing 242
- REJECT command operand 259
- REJECT processing 272
- RESTORE processing 338

**HOLDDATA entry**

- listing 219, 695
- saving after ACCEPT processing (NOPURGE) 743
- summary 695

**HOLDERROR**

- BYPASS operand
  - ACCEPT command 8
  - ACCEPT processing 33
  - APPLY command 56
  - APPLY processing 81
- LIST command operand 220

**HOLDSYSTEM**

- BYPASS operand
  - ACCEPT command 8
  - ACCEPT processing 33
  - APPLY command 56
  - APPLY processing 81
- LIST command operand 221

**HOLDUSER**

- BYPASS operand
  - ACCEPT command 10
  - ACCEPT processing 33
  - APPLY command 57
  - APPLY processing 81
- LIST command operand 221

**I**

**I/O GEN**

- related to JCLIN 172

**ID**

- BYPASS operand
  - ACCEPT command 10
  - APPLY command 58
  - RESTORE command 333

**IDCAMS utility**

*See* AMS utility

**IDENTIFY statement**

JCLIN processing 194

**IEANUC01**

*See also* NUCID

APPLY processing 744

RESTORE considerations 338

saving

APPLY processing 100

LINK processing 208

**IEBCOPY utility**

*See* copy utility

**IEBUPDTE utility**

*See* update utility

**IEHIOSUP utility**

*See also* VS1 installation utility

OPTIONS entry 743

UTILITY entry for 743

**IEV90 utility**

*See* assembler utility

**IEWBLINK utility**

*See* link-edit utility

**IEWL utility**

*See* link-edit utility

**IF**

MCS statement 546

ZONEEDIT command operand 408

**IFREQ**

*See also* ++IF MCS

*See also* conditional requisites

ACCEPT processing 32

APPLY processing 80

BYPASS operand

ACCEPT command 10

APPLY command 58

SYSMOD entry

distribution zone 371, 761

target zone 371, 761

**IHASUxx**

consideration for deleting a function 36

**IMASPZAP utility**

*See* superzap utility

**implicitly deleting functions**

ACCEPT processing 35

APPLY processing 83

**implicitly including modules from another product 196, 201****IMPORT**

*See also* AMS utility

*See also* ZONEIMPORT command

after conversion 122

**INCLUDE statement**

JCLIN processing 194

**incompatibility with previous releases of SMP/E 865****INDEX**

ZONEEXPORT command operand 414

**index for online library**

MCS for

*See* data element MCS

**INFILE**

defining 607

ZONEIMPORT command operand 418

**inline elements 528, 539, 556, 567, 584****inline JCLIN**

ACCEPT processing 36

adding new load modules 68

APPLY processing 85

JCLIN processing 167, 170, 180, 181

packaging 171, 550

RESTORE processing 342

**input for utility programs**

MCS for

*See* data element MCS

**INSERT statement**

JCLIN processing 194

**installation exits**

*See* installation-wide exit routines for SMP/E

**installation job streams, generating with the GENERATE command 149****installation-wide exit routines**

*See* installation-wide exit routines for SMP/E

**installation-wide exit routines for SMP/E 797, 798**

activating 798

GIMMPUXD 798

parameter list mapping 797, 798

RECEIVE exit 246

RECEIVE exit (Exit 1) 799

Retry Exit (Exit 2) 801

summary 797

**INSTALLDATE**

SYSMOD entry

distribution zone 371, 762

target zone 371, 762

**installing SYSMODs**

*See* ACCEPT command

*See* APPLY command

*See* GENERATE command

**INSTALLTIME**

SYSMOD entry

distribution zone 371, 762

target zone 371, 762

**instream procedures**

not recognized in JCLIN 181

**internal HOLDDATA 486, 540, 578, 695****INTO**

ZONECOPY command operand 391

ZONEIMPORT command operand 418

## INTO *(continued)*

ZONEMERGE command operand 424

## INTOLMOD

LINK command operand 202

## invoking SMP/E 813

## INZONE

REPORT SYSMODS command operand 317

## IOGEN reason ID 9, 57, 542, 577, 808

## IOSUP

OPTIONS entry 369, 743

## ISPF dialogs, packaging

MCS for

*See* data element MCS

## J

### JCL generated by GENERATE command 160

#### JCLIN

*See also* ++JCLIN MCS

*See also* JCLIN command

inline

ACCEPT processing 36

APPLY processing 85

RESTORE processing 342

to add new load modules 68

packaged inline 549

PARMLIB OP CODE members

operands 604

summary 603

syntax 604

SYSMOD entry

distribution zone 371, 762

global zone 775

target zone 371, 762

#### JCLIN command

assembler steps 171

coding conventions

assembler 183

copy 185

examples 181, 182

link-edit 188

other 198

summary 180

update 198

cross-zone relationships 172, 198

data set sharing 199

data sets required 170

determining macros 184

ENQ considerations 199

inline JCLIN 171

instream procedures not recognized 181

operands

ASM 168

CALLLIBS 168

COPY 168

JCLINREPORT 168

#### JCLIN command *(continued)*

operands *(continued)*

LKED 169

NOJCLINREPORT 169

OPCODE 169

PGM 169

RC 169

UPDATE 169

processing 179

processing assembly steps

creating ASSEM entry 183

creating MAC entry 184

creating SRC entry 184

processing copy steps

creating DLIB entry 186

creating LMOD entry 187

creating MOD entry 187

summary 185

processing link-edit steps

coding conventions 188

creating LMOD entry 195

creating MOD entry 194

processing other steps 198

processing update steps 198

sample input 174

summary 167

syntax 168

SYSMOD with inline JCLIN 170

system generation 172

usage notes 170

zone for SET BOUNDARY 168

#### JCLIN reports

*See also* JCLINREPORT

*See also* NOJCLINREPORT

Cross-Reference report 471

summary report 473

#### JCLINREPORT

ACCEPT command operand 14

APPLY command operand 62

JCLIN command operand 168

#### job card

*See* JOBCARD

#### JOBCARD

GENERATE command operand 150

REPORT CALLLIBS command operand 275

## K

### KEEP

DDDEF entry 361, 655

## L

### language abbreviations 525



**LANGUAGE parameter, EXEC statement for****GIMSMP** 814**LAST UPDATE**

See LASTUPD

**LAST UPDATE TYPE**

See LASTUPDTYPE

**LASTSUP**

SYSMOD entry

distribution zone 371, 762

target zone 371, 762

**LASTUPD**

ASSEM entry 359, 640

data element entry 360, 649

DLIB entry 363, 671

HFS entry 365, 688

LMOD entry 366, 699

MAC entry 367, 716

MOD entry 368, 724

SRC entry 370, 750

SYSMOD entry

distribution zone 371, 762

target zone 371, 762

**LASTUPDTYPE**

ASSEM entry 359, 641

data element entry 360, 649

DLIB entry 363, 671

HFS entry 365, 688

LMOD entry 366, 699

MAC entry 367, 716

MOD entry 368, 725

SRC entry 370, 751

SYSMOD entry

distribution zone 371, 762

target zone 371, 762

**LDELETE**

SYSMOD entry

distribution zone 371

target zone 371

**LEPARM**

++MOD MCS operand 565

ACCEPT processing 46

APPLY processing 98

example of use 568

**level of SMP/E** 441**LIBRARY statement**

JCLIN processing 195

**LIBRARYDD comment for pathname in link-edit steps** 196, 197**LINK**

See also LINK command

++HFS MCS operand 536

**LINK command**

data set sharing 209

data sets required 203

ENQ considerations 209

**LINK command** (*continued*)

example 204

HFS entry 365, 688

operands

FROMZONE 202

INTOLMOD 202

MODULE 202

RC 203

RETRY 203

processing 205

reports 204

restriction on using zone with previous releases of SMP/E 201

restrictions 866

summary 201

syntax 202

zone for SET BOUNDARY 202

**link library**

See LKLIB

**link-edit parameters**

See LEPARM

**link-edit utility**

ACCEPT processing 45, 46

APPLY processing 97

default values 743, 787, 789, 836

effect of LEPARM 568

GENERATE processing 162

JCLIN processing 188

LINK processing 208

OPTIONS entry 369, 743

parameters

ACCEPT processing 46

APPLY processing 97

recognized by SMP/E 197

parameters recognized by SMP/E 197

specifying on JCLIN 169, 548

UTILITY entry for 743, 787

**linking modules from another zone** 201**linknames**

See LINK

**LIST**

See also LIST command

RECEIVE command operand 240

UTILITY entry 788

restrictions 866

**LIST command**

data set sharing 234

data sets required 230

ENQ considerations 234

examples 231

modes of processing

mass mode 234

select mode 234

operands

ALLZONES 215

APARS 216

**LIST command** *(continued)*operands *(continued)*

ASSEM 216  
BACKUP 216  
BYPASS 216  
DDDEF 216  
DELETE 216  
DLIB 217  
DLIBZONE 217  
element 217  
ERROR 217  
EXSRCID 217  
FMIDSET 218  
FORFMID 218  
FUNCTIONS 219  
GLOBALZONE 219  
HFS 219  
HOLDDATA 219  
HOLDERROR 220  
HOLDSYSTEM 221  
HOLDUSER 221  
LMOD 221  
LOG 221  
MAC 222  
MCS 222  
MOD 222  
NOACCEPT 222  
NOAPPLY 223  
NOSUP 224  
OPTIONS 224  
PTFS 224  
RESTORE 225  
SOURCEID 225  
SRC 225  
SUP 226  
SYSMODS 226  
TARGETZONE 227  
USERMODS 227  
UTILITY 227  
XREF 228  
XZLMODP 228  
XZMODP 228  
ZONESET 229  
processing 233  
reports 230, 477  
summary 211  
summary report 477  
syntax 212  
syntax notes 229  
usage notes 230  
zone for SET BOUNDARY 211

**listing cover letters** 222**LKED**

*See also* link-edit utility  
++JCLIN MCS operand 548

**LKED** *(continued)*

JCLIN command operand 169  
JCLIN processing 180, 188  
OPTIONS entry 369, 743  
UTILITY entry for 787

**LKED ATTRIBUTES**

*See also* LEPARM  
LMOD entry 699  
MOD entry 725

**LKED CONTROL**

LMOD entry 702

**LKLIB**

++MOD MCS operand 565  
defining 608

**lklib data set**

*See* LKLIB

**LKSYSLIB job, built by GENERATE to link-edit load modules having a SYSLIB concatenation** 163**LMOD**

*See also* LMOD entry  
++MOD MCS operand 565  
++MOD statement 68  
++MOVE MCS operand 570  
CONVERT command operand 116  
LIST command operand 221  
MOD entry 727  
UNLOAD command operand 383

**LMOD entry**

*See also* load modules  
created by JCLIN 187, 195  
listing 221, 703  
restriction on CALLLIBS subentry 196, 698  
subentries  
CALLLIBS 698  
COPY 698  
LASTUPD 699  
LASTUPDTYPE 699  
LKED ATTRIBUTES 699  
LKED CONTROL 702  
MODDEL 702  
SYSLIB 702  
XZMOD 703  
XZMODP 703

summary 698  
UCLIN for 366, 710  
unloading 383, 707  
updating cross-zone subentries  
ZONEEDIT command 405

**load module attributes**

*See* LEPARM

**load modules**

*See also* cross-zone load modules  
*See also* LMOD entry  
adding new 68  
aliases for 566

**load modules (continued)**

- attributes
  - ACCEPT processing 46
  - APPLY processing 97
- building 90
- building with a SYSLIB allocation 99
- deleting 84, 86, 529
- hierarchical file system (HFS)
  - JCLIN for 178
  - LIBRARYDD comment in link-edit steps 196, 197
  - SYSLIB DD statement in link-edit steps 196
  - SYSLMOD DD statement in link-edit steps 197
- linking modules from another zone 201
- moving 86, 570
- renaming 86, 580
- updating 600

**loading SYSMODs from the distribution**

medium 239

**loading system utility programs 835****local shared resources (LSR) 609, 625, 867**

*See also* batch local shared resources (LSR), performance improved by

**LOG**

- See also* LOG command
- See also* SMPLOG
- LIST command operand 221

**LOG command**

- data sets required 236
- examples 236
- operands 235
- processing 237
- reports 236
- summary 235
- syntax 235
- zones for SET BOUNDARY 235

**LSR**

*See* batch local shared resources (LSR), performance improved by

**LVL value of SMP/E 441****M****MAC**

- See also* MAC entry
- See also* macros
- ++MOVE MCS operand 570
- CONVERT command operand 116
- LIST command operand 222
- MCS statement 553
- PARMLIB statement 604
- SYSMOD entry
  - distribution zone 371, 763
  - global zone 775
  - target zone 371, 763

**MAC (continued)**

UNLOAD command operand 383

**MAC entry**

- See also* macros
- ++MAC MCS 553
- created by JCLIN 184
- listing 222, 717
- subentries
  - DISTLIB 715
  - FMID 715
  - GENASM 716
  - LASTUPD 716
  - LASTUPDTYPE 716
  - MALIAS 716
  - RMID 716
  - SYSLIB 716
  - UCL syntax 715
  - UMID 717
- summary 715
- UCLIN for 367, 719
- unloading 383, 718

**macros**

- See also* MAC entry
- See also* MTSMAC entry
- ++MAC MCS 554
- ++MACUPD MCS 559
- adding 553
- assemblies caused by
  - ACCEPT processing 44
  - APPLY processing 96
- defining 553
- deleting
  - ++MAC MCS 554
  - ACCEPT processing 35
  - APPLY processing 83, 91
- MCS statement 554, 559
- moving 570
- replacing
  - ++MAC 553
  - ACCEPT processing 42
  - APPLY processing 92
- updating
  - ++MACUPD MCS 559
  - ACCEPT processing 42
  - APPLY processing 93

**macros intended for customer use 879****MACUPD**

- MCS statement 559
- SYSMOD entry
  - distribution zone 371, 763
  - global zone 776
  - target zone 371, 763

**MALIAS**

- ++MAC MCS operand 554
- ++MACUPD MCS operand 560

## **MALIAS** *(continued)*

- ACCEPT processing 20
- APPLY processing 70
- MAC entry 367, 716
- RECEIVE processing 251

## **mass-mode processing**

- ACCEPT command 30
- APPLY command 78
- LIST command 234
- RECEIVE command 251
- REJECT command 255, 268
- UNLOAD command 389

## **master CSI**

- specified on DD statement 815
- specified on EXEC statement 813, 815
- use 609

## **MAXBLK**

- LMOD entry 700
- MOD entry 726

## **MCS**

- See also* MCS entry
- See also* MCS statements
- CONVERT command operand 116
- LIST command operand 222

## **MCS entry**

- listing 222, 721
- subentries 721
- summary 721

## **MCS statements**

- See also* MCS entry
- ++APAR 519
- ++ASSIGN 521
- ++DELETE 529
- ++FUNCTION 533
- ++HFS 535
- ++HOLD 540
- ++IF 546
- ++JCLIN 548
- ++MAC 553
- ++MACUPD 559
- ++MOD 563
- ++MOVE 570
- ++NULL 573
- ++PTF 574
- ++RELEASE 576
- ++RENAME 580
- ++SRC 582
- ++SRCUPD 586
- ++USERMOD 589
- ++VER 591
- ++ZAP 598
- data element 523
- general syntax rules 2
- overview 517

## **merging**

- CSI data sets 423
- zones 423

## **messages for dialogs**

- MCS for
- See* data element MCS

## **messages, tracing 143**

## **MGMTCLAS**

- DDDEF entry 657

## **migrating from previous releases of SMP/E 865**

## **missing SYSMODs**

- checking for
- REPORT SYSMODS command 317

## **MOD**

- See also* ++MOD MCS
- See also* MOD entry
- See also* modules
- ++MOVE MCS operand 570
- CONVERT command operand 116
- DDDEF entry 361, 657
- LIST command operand 222
- MCS statement 563
- SYSMOD entry
  - distribution zone 371, 763
  - global zone 776
  - target zone 371, 763
- UNLOAD command operand 383

## **MOD entry**

- See also* modules
- ++MOD MCS 563
- created by JCLIN 187, 194
- listing 222, 729
- subentries
  - ASSEMBLE 723
  - CSECT 723
  - DALIAS 724
  - DISTLIB 724
  - FMID 724
  - LASTUPD 724
  - LASTUPDTYPE 725
  - LKED ATTRIBUTES 725
  - LMOD 727
  - RMID 728
  - RMIDASM 728
  - TALIAS 728
  - UMID 728
  - XZLMOD 728
  - XZLMODP 729
- summary 723
- UCLIN for 368, 736
- unloading 383, 733
- updating cross-zone subentries
  - ZONEEDIT command 405

## **MODEL**

- LMOD entry 702

**modes of processing**

- ACCEPT command
  - mass mode 30
  - select mode 30
- APPLY command
  - mass mode 78
  - select mode 78
- LIST command
  - mass mode 234
  - select mode 234
- mass mode
  - ACCEPT command 30
  - APPLY command 78
  - RECEIVE command 251
  - REJECT command 255, 268
- NOFMID mode
  - REJECT command 255, 271
- PURGE mode
  - REJECT command 255, 269
- RECEIVE command
  - mass mode 251
  - select mode 251
- REJECT command
  - mass mode 268
  - NOFMID mode 271
  - PURGE mode 269
  - select mode 269
- RESTORE command
  - group mode 337
  - select mode 337
- select mode
  - ACCEPT command 30
  - APPLY command 78
  - RECEIVE command 251
  - REJECT command 255, 269
- UNLOAD command
  - mass mode 389
  - select mode 389

**MODIDs**

- See* FMID
- See* RMID
- See* UMID

**modification control statement**

- See* MCS

**modification level**

- See* FMID
- See* RMID
- See* UMID

**MODULE**

- LINK command operand 202

**modules**

- See also* cross-zone modules
- See also* MOD entry
- ++MOD MCS 563
- ++ZAP MCS 598

**modules (continued)**

- adding 563
- defining 563
- deleting
  - ++MOD MCS 564
  - ACCEPT processing 35
  - APPLY processing 84, 91
- linking from another zone 201
- moving 570
- reintroducing with MODDEL subentries 84
- replacing
  - ++MOD MCS 563
  - ACCEPT processing 45
  - APPLY processing 97
- updating
  - ++ZAP MCS 598
  - ACCEPT processing 46
  - APPLY processing 100

**MOVE**

- See also* ++MOVE MCS
- SYSMOD entry
  - distribution zone 371
  - target zone 371

**MOVE/RENAME/DELETE report 479****moving elements**

- ++MOVE MCS 570
- ACCEPT processing 37
- APPLY processing 86
- RESTORE processing 345

**moving load modules**

- ++MOVE MCS 570
- RESTORE processing 345

**MSGMODID**

- DEBUG command operand 145

**MSGSKEL reason ID 9, 57, 542, 577, 808****MTS**

- See* SMPMTS

**MTSMAC entry**

- ACCEPT processing 48
- RESTORE processing 344
- saving through the OPTIONS entry 745
- subentries 739
- summary 739
- UCLIN for 369, 739

**multiple-CSI structure**

- converting 127

**MVSCP reason ID 9, 57, 543, 578, 808****N****NAME**

- ++DELETE MCS operand 530
- ++HFS MCS operand 537
- ++MAC MCS operand 555
- ++MACUPD MCS operand 560

## **NAME** *(continued)*

- ++MOD MCS operand 566
- ++MOVE MCS operand 570
- ++SRC MCS operand 583
- ++SRCUPD MCS operand 586
- ++ZAP MCS operand 598
- data element MCS operand 527
- UTILITY entry 375, 788
- ZONEMERGE command operand 424

## **NAME statement**

- JCLIN processing 195

## **naming conventions**

- CLASS 808
- HOLD reason IDs 807
- reason IDs 807
- SOURCEIDs 808
- summary 805
- SYSMOD IDs
  - APARs 810
  - FUNCTIONS 810
  - PTFs 810
  - summary 809
  - USERMODs 810
- zones 675, 782

## **national language identifiers 525**

## **national language support**

- See* NLS (national language support)

## **NCAL**

- passed based on existence of CALLLIBS
  - subentry 98, 787, 789
- used in GENERATE jobs for load modules with CALLLIBS 159

## **NE**

- LMOD entry 366, 700
- MOD entry 368, 726

## **negative prerequisite SYSMODs**

- ACCEPT processing 32
- APPLY processing 81
- defining with the NPRES operand 592

## **negative prerequisites**

- See* negative prerequisite SYSMODs
- See* NPRES

## **NEW**

- DDDEF entry 361, 657

## **NEWCSI**

- CONVERT command operand 116
- operands
  - NEWCSI 116

## **NEWDATASET**

- ZONERENAME command operand 434

## **NLS (national language support)**

- language abbreviations 525

## **NOACCEPT**

- LIST command operand 222
- UNLOAD command operand 383

## **NOAPPLY**

- LIST command operand 223
- UNLOAD command operand 383

## **NOCALL**

- LMOD entry 701
- MOD entry 726

## **NOFMID**

- REJECT command operand 260
- REJECT processing 255, 271

## **NOFMID mode processing**

- REJECT command 255, 271

## **NOJCLIN**

- ACCEPT command operand 14
- ACCEPT processing 37
- APPLY command operand 62
- APPLY processing 86

## **NOJCLINREPORT**

- ACCEPT command operand 14
- APPLY command operand 62
- JCLIN command operand 169

## **NOPUNCH**

- REPORT CALLLIBS command operand 276
- REPORT CALLLIBS processing 284
- REPORT CROSSZONE command operand 288
- REPORT CROSSZONE processing 296
- REPORT ERRSYSMODS command operand 301
- REPORT ERRSYSMODS processing 305
- REPORT SOURCEID command operand 309
- REPORT SOURCEID processing 315
- REPORT SYSMODS command operand 318
- REPORT SYSMODS processing 325

## **NOPURGE**

- default value 744
- OPTIONS entry 369, 743
  - ACCEPT processing 50
- ZONEEXPORT command operand 414

## **NOREJECT**

- default value 744
- OPTIONS entry 369, 744
- RESTORE processing 338, 347

## **NOREPLACE**

- ZONEMERGE command operand 424
- ZONEMERGE processing 429

## **NOSUP**

- LIST command operand 224
- UNLOAD command operand 384

## **NPRES**

- See also* negative prerequisite SYSMODs
- ++VER MCS operand 592
- ACCEPT processing 32
- APPLY processing 81
- SYSMOD entry
  - distribution zone 371, 763
  - global zone 776
  - target zone 371, 763

**NUCID**

- APPLY command operand 62
- APPLY processing 100
- LINK processing 208
- OPTIONS entry 369, 744

**nucleus load module**

- See IEANUC01
- See NUCID

**NULL**

- MCS statement 573

**NULLFILE used for dummy data sets**

- defined in DDDEF entry 656
- not allowed for SMPPTS 615

**O****object elements for printers**

- MCS for
- See data element MCS

**object modules**

- See modules

**OL**

- LMOD entry 701
- MOD entry 726

**OLD**

- DDDEF entry 361, 657

**OLDCSI**

- CONVERT command operand 116

**online publications library**

- MCS for
- See data element MCS

**OPCODE**

- ++JCLIN MCS operand 549
- defined in PARMLIB 603
- JCLIN command operand 169
- JCLIN processing 171, 184
- PARMLIB statement 604

**OPTIONS**

- See also OPTIONS entry
- DLIBZONE entry 363, 675
- GLOBALZONE entry 364, 682
- LIST command operand 224
- SET command operand 349, 352
- TARGETZONE entry 374, 782
- ZONECOPY command operand 392
- ZONEIMPORT command operand 418
- ZONERENAME command operand 434

**OPTIONS entry**

- defining default
  - distribution zone 675
  - global zone 682
  - target zone 782
- GENERATE processing 160
- listing 224, 746
- overriding default with SET command 352

**OPTIONS entry (continued)**

- specified on SET command 349
- subentries
  - AMS 741
  - ASM 741
  - COMP 741
  - COPY 742
  - DSPREFIX 742
  - DSSPACE 742
  - EXRTYDD 742
  - HFSCOPY 743
  - IOSUP 743
  - LKED 743
  - NOPURGE 743
  - NOREJECT 744
  - NUCID 744
  - PAGELEN 744
  - PEMAX 744
  - RETRY 744
  - RETRYDDN 745
  - SAVEMTS 745
  - SAVESTS 745
  - UPDATE 745
  - ZAP 746
- summary 741
- UCLIN for 369, 747

**ORDER statement**

- JCLIN processing 195

**organization of SMP/E data sets 627****out-of-space errors**

- See COMPRESS
- See PEMAX
- See RETRY

**OUTFILE**

- defining 608
- ZONEEXPORT command operand 413

**output for utility programs**

- MCS for
- See data element MCS

**OVERLAY statement**

- JCLIN processing 194

**OVLY**

- LMOD entry 366, 701
- MOD entry 368, 726

**P****packaging SYSMODs**

- indirect libraries
  - LKLIB 565
  - TXLIB 538, 549, 556, 566, 583
- inline JCLIN 171
- LKLIB 565
- relative files (RELFILES)
  - FILES operand on ++APAR 519
  - FILES operand on ++FUNCTION 533

**packaging SYSMODs** *(continued)*

- relative files (RELFILES) *(continued)*
  - FILES operand on ++HFS 538
  - FILES operand on ++MAC 555
  - FILES operand on ++USERMOD 589
- RECEIVE processing 243
- reference material 533
- REJECT processing 272
- RELFILE operand on ++JCLIN 549
- RELFILE operand on ++MOD 566
- RELFILE operand on ++PTF 574
- RELFILE operand on ++SRC 583
- RFDSNPFX operand on ++APAR 520
- RFDSNPFX operand on ++FUNCTION 534
- RFDSNPFX operand on ++PTF 574
- RFDSNPFX operand on ++USERMOD 590
- TXLIB 538, 549, 556, 566, 583

**PAGELEN**

- default value 744
- OPTIONS entry 369, 744

**panels for dialogs**

- MCS for
  - See data element MCS

**PARM**

- ++HFS MCS operand 537
- HFS entry 688
- UTILITY entry 375, 788

**PARMLIB**

- See also PARMLIB members
- defining 608

**PARMLIB members**

- JCLIN OPCODEs 603
- MCS for
  - See data element MCS
- operands 604
- syntax 604
- syntax rules 2

**PATH**

- DDDEF entry 363, 658
- operand for HFS pathname 196, 197

**pathname, DDDEF entry for 658****PE-PTFs**

- See also ERROR
- See also exception SYSMODs
- checking for
  - REPORT ERRSYSMODS command 299

**PEMAX**

- default value 744
- OPTIONS entry 369, 744

**pending updates to zones 834****performance**

- BUFND parameter 867
- BUFNI parameter 867
- improvements
  - through batch local shared resources (LSR) sub-system 867

**performance** *(continued)*

- local shared resources (LSR) feature of VSAM 609, 625, 867

**PGM**

- JCLIN command operand 169

**PRE**

- See also prerequisite SYSMODs
- ++VER MCS operand 592
- ACCEPT processing 32
- APPLY processing 80
- BYPASS operand
  - ACCEPT command 10
  - APPLY command 58
- SYSMOD entry
  - distribution zone 371, 763
  - global zone 776
  - target zone 371, 763

**PREFIX**

- ++MAC MCS operand 555
- ++MACUPD MCS operand 560

**prefix for relative file data sets**

- See RFDSNPFX

**prerequisite SYSMODs**

- See also NPRES
- See also PRE
- ++IF MCS 546
- ACCEPT processing 32
- APPLY processing 80
- defining 592

**PRINT**

- See also SYSPRINT
- UTILITY entry 375, 789

**printer object elements**

- MCS for
  - See data element MCS

**printer source elements**

- MCS for
  - See data element MCS

**printing cover letters 222****problems, debugging 143****procedures**

- MCS for
  - See data element MCS

**PROCESS parameter, EXEC statement for GIMSMP 814****PROCESS=WAITIEND**

- See EXEC statement

**PROCLIB members**

- MCS for
  - See data element MCS

**product release specified in FMID 810****PROTECT**

- DDDEF entry 361, 658

**PSEG for online book**

- MCS for
  - See data element MCS



**PTF**

ACCEPT command operand 14  
 ACCEPT processing 29  
 APPLY command operand 62  
 APPLY processing 78  
 defining 574  
 LIST command operand 224  
 MCS statement 574  
 naming conventions 810  
 REJECT command operand 260  
 REJECT processing 268, 269  
 SYSMOD entry  
   distribution zone 371, 763  
   global zone 776  
   target zone 371, 763  
 UNLOAD command operand 384

**PTS**

*See also* SMPPTS  
 CONVERT command operand 117  
 converting 113  
 SYSTEM entry conversion 140

**publications library**

MCS for  
   *See* data element MCS

**PURGE**

REJECT command operand 260  
 REJECT processing 269  
 ZONEEXPORT command operand 414  
 ZONEEXPORT processing 416

**PURGE mode processing**

REJECT command 255, 269

**PUTyynn**

SOURCEID  
   naming conventions 809

**R****RACF protection for data sets 658****RC**

*See also* return codes  
 ACCEPT command operand 14  
 APPLY command operand 62  
 CLEANUP command operand 108  
 CONVERT command operand 117  
 explanation 829  
 GENERATE command operand 150  
 JCLIN command operand 169  
 LINK command operand 203  
 RECEIVE command operand 240  
 REJECT command operand 261  
 RESTORE command operand 334  
 SMP/E default threshold 829  
 UCLIN command operand 357  
 UTILITY entry 375, 790  
 ZONECOPY command operand 392

**RC (continued)**

ZONEDELETE command operand 399  
 ZONEEDIT command operand 406  
 ZONEEXPORT command operand 414  
 ZONEIMPORT command operand 418  
 ZONEMERGE command operand 424  
 ZONERENAME command operand 434

**reaccepting SYSMODs 23, 40****reading in SYSMODs from the distribution medium 239****reapplying SYSMODs 73, 80, 90****REASON**

++HOLD MCS operand 542  
 ++RELEASE MCS operand 577

**reason IDs**

ACCEPT processing 32  
 APPLY processing 81  
 naming conventions 807  
 RECEIVE processing 253

**RECDATE**

SYSMOD entry  
   distribution zone 371, 763  
   global zone 776  
   target zone 371, 763

**RECEIVE command 239**

data set sharing 254  
 data sets required 242  
 ENQ considerations 254  
 installation-wide exit routine 246  
 modes of processing  
   mass mode 251  
   select mode 251

**operands**

BYPASS 239  
 EXCLUDE 239  
 FORFMID 239  
 HOLDDATA 240  
 LIST 240  
 RC 240  
 RFPREFIX 241  
 SELECT 241  
 SOURCEID 242  
 SYSMODs 242

**output**

listings 246  
 reports 247  
 processing 249  
 summary 239  
 summary report 488  
 syntax 239  
 syntax notes 242

**SYSMOD selection**

effect of GLOBALZONE FMID list 251  
 effect of GLOBALZONE SREL list 251  
 EXCLUDE operand 251  
 SELECT operand 251

**RECEIVE command** *(continued)*

zone for SET BOUNDARY 239

**RECEIVE Exception SYSMOD Data report** 485**RECEIVE exit routine** 799**RECTIME**

SYSMOD entry

distribution zone 371, 763

global zone 776

target zone 371, 763

**REDO**

ACCEPT command operand 15

ACCEPT processing 31

APPLY command operand 63

APPLY processing 80

**REFR**

LMOD entry 366, 701

MOD entry 368, 727

**REGEN**

ACCEPT processing 49

SYSMOD entry

distribution zone 371, 764

target zone 371, 764

**region size for SMP/E** 821**regression**

APPLY processing 105

reports 503

**reinstalling products by using GENERATE** 153**reinstalling products without SYSGEN support****(GENERATE command)** 149**REJECT command**

data set sharing 272

data sets required 108, 262

ENQ considerations 272

examples 264

modes of processing

mass mode 255, 268

NOFMID mode 255, 271

PURGE mode 255, 269

select mode 255, 269

operands

APARS 257

BYPASS 257

COMPRESS 258

DELETFMID 258

EXCLUDE 258

EXCLUDEZONE 258

FORFMID 259

FUNCTIONS 259

HOLDDATA 259

NOFMID 260

PTFS 260

PURGE 260

RC 261

SELECT 261

SOURCEID 261

TARGETZONE 262

**REJECT command** *(continued)*operands *(continued)*

USERMODS 262

output

reports 263

statistics 263

processing

mass mode 268

NOFMID mode 271

PURGE mode 269

select mode 269

summary 267

summary 255

summary report 492

syntax 255

zone for SET BOUNDARY 255

**RELATED***See also* related zone

DLIBZONE entry 363, 675

TARGETZONE entry 374, 782

ZONECOPY command operand 392

ZONEIMPORT command operand 418

ZONERENAME command operand 435

**related zone***See also* RELATED

defining

for a copied zone 392

for a distribution zone 675

for a renamed zone 435

for a target zone 782

for an imported zone 418

summary 632

**relative files (RELFILES)***See also* RELFILE*See also* SMPTLIB

deletion of associated SMPTLIB data sets

ACCEPT processing 50

RECEIVE processing 243

REJECT processing 272

RESTORE processing 347

prefix for 520

RECEIVE processing 249

**RELEASE***See also* ++RELEASE MCS

specified in FMID 810

**release summary** 839**releasing a held SYSMOD**

++RELEASE MCS 576

**RELFILE**

++HFS MCS operand 538

++JCLIN MCS operand 549

++MAC MCS operand 555

++MOD MCS operand 566

++SRC MCS operand 583

data element MCS operand 527

**RELFILE format**

See relative files (RELFILES)

See SMPTLIB

**removing changes from the target libraries**

(RESTORE command) 333

**removing SYSMODs from target libraries (RESTORE command) 333****RENAME**

See ++RENAME MCS

**renaming load modules**

++RENAME MCS 580

APPLY processing 86

**renaming zones 433****RENLMOD**

SYSMOD entry

distribution zone 371, 764

target zone 371, 764

**RENT**

LMOD entry 366, 701

MOD entry 368, 727

**REP UCL statement 356****REPLACE**

GENERATE command operand 150

ZONEMERGE command operand 425

ZONEMERGE processing 429

**REPLACE statement**

JCLIN processing 195

**replacing data elements**

See data element MCS

**replacing data set entries via UCL statements 357****REPORT CALLLIBS command**

data set sharing 286

data sets required 276

ENQ considerations 286

example 279

operands

CALLLIBS 275

JOB CARD 275

NOPUNCH 276

ZONES 276

output

reports 277

SMPPUNCH 277

processing 283

summary 275

syntax 275

zone for SET BOUNDARY 275

**REPORT CROSSZONE command**

Cross-Zone Requisite SYSMOD report 447

data set sharing 297

data sets required 289

ENQ considerations 297

example with zones controlled by different global zones 294

example with zones controlled by the same global zone 291

**REPORT CROSSZONE command (continued)**

operands

CROSSZONE 287

DLIBZONE 287

FORFMID 288

FORZONE 288

NOPUNCH 288

TARGETZONE 288

ZONESET 289

output

reports 289

SMPPUNCH 290

processing 295

reports 447

summary 287

syntax 287

usage notes 289

zone for SET BOUNDARY 287

**REPORT ERRSYSMODS command**

data set sharing 307

data sets required 301

ENQ considerations 307

example 304

Exception SYSMOD report 460

operands

BEGINDATE 299

ENDDATE 300

ERRSYSMODS 300

FORFMID 301

NOPUNCH 301

ZONES 301

output

reports 302

SMPPUNCH 302

processing 305

reports 460

summary 299

syntax 299

usage notes 302

zone for SET BOUNDARY 299

**REPORT SOURCEID command**

data set sharing 315

data sets required 310

ENQ considerations 315

examples 311

operands

NOPUNCH 309

SOURCEID 309

SYSMODIDS 309

ZONES 310

output

reports 310

SMPPUNCH 310

processing 314

reports 498

**REPORT SOURCEID command** *(continued)*

- SOURCEID report 498
- summary 309
- syntax 309
- zone for SET BOUNDARY 309

**REPORT SYSMODS command**

- data set sharing 328
- data sets required 318
- ENQ considerations 328
- example 321
- operands
  - COMPAREDTO 317
  - INZONE 317
  - NOPUNCH 318
  - SYSMODS 318
- output
  - reports 318
  - SMPPUNCH 318
- processing 325
- reports 500
- summary 317
- syntax 317
- SYSMOD Comparison report 500
- zone for SET BOUNDARY 317

**reports**

- CALLLIBS Summary report 442
- Causer SYSMOD Summary report 444
- CLEANUP Summary report 446
- Cross-Zone Requisite SYSMOD report 447
- Cross-Zone Summary report 449
- Deleted SYSMOD report 453
- description 441
- Element Summary report 455
- Exception SYSMOD report 460
- File Allocation report 463
- GENERATE Summary report 467
- JCLIN Cross-Reference report 471
- JCLIN Summary report 473
- LIST Summary report 477
- MOVE/RENAME/DELETE report 479
- RECEIVE Exception SYSMOD Data report 485
- RECEIVE Summary report 488
- REJECT Summary report 492
- SOURCEID report 498
- summary 441
- SYSMOD Comparison report 500
- SYSMOD Regression report 503
- SYSMOD Status report 505
- UNLOAD Summary report 508
- ZONEEDIT Summary report 510
- ZONEMERGE report 512

**REPRO**

- See also* AMS utility
- merging CSIs 423

**REQ**

- See also* corequisite SYSMODs
- ++IF MCS operand 546
- ++VER MCS operand 592
- ACCEPT processing 32
- APPLY processing 80
- BYPASS operand
  - ACCEPT command 10
  - APPLY command 58
- SYSMOD entry
  - distribution zone 371, 764
  - global zone 776
  - target zone 371, 764

**requisite SYSMODs**

- See also* conditional requisites
- See also* corequisite SYSMODs
- See also* negative prerequisite SYSMODs
- See also* prerequisite SYSMODs
- ACCEPT processing 32
- APPLY processing 80

**rereceiving SYSMODs 252**

**RESDATE**

- SYSMOD entry
  - target zone 371, 764

**RESETRC command**

- data sets required 329
- examples 330
- processing 331
- summary 329
- syntax 329
- usage notes 329
- zone for SET BOUNDARY 329

**resetting SMP/E return codes 329**

**resolving held SYSMODs (REPORT ERRSYSMODS command) 299**

**resolving SYSMODs, checking for (REPORT ERRSYSMODS command) 299**

**RESTIME**

- SYSMOD entry
  - target zone 371, 764

**RESTORE**

- See also* RESTORE command
- LIST command operand 225
- SYSMOD entry
  - target zone 371, 764
- UNLOAD command operand 384

**RESTORE command**

- cross-zone processing 346
- data set sharing 347
- data sets required 335
- deleted elements 343
- deleted load modules 345
- element installation
  - assemblies 344
  - data elements 344
  - HFS elements 344

**RESTORE command** *(continued)*

element installation *(continued)*  
   macros 344  
   modules 344  
   source 344  
   summary 342  
 ENQ considerations 347  
 examples 339  
 inline JCLIN processing 342  
 load modules created by the SYSMOD being restored 345  
 load modules with a SYSLIB allocation 345  
 modes of processing  
   group mode 337  
   select mode 337  
 moved elements 345  
 operands  
   BYPASS 333  
   CHECK 333  
   COMPRESS 334  
   GROUP 334  
   RC 334  
   RETRY 335  
   SELECT 335  
 processing  
   compress 343  
   summary 341  
 renamed load modules 345  
 reports 338  
 SMPLTS cleaned up 345  
 summary 333  
 syntax 333  
 SYSMOD selection  
   operands 341  
 updating the SMPSCDS BACKUP entries 346  
 updating the target zone entries 346  
 updating the target zone SYSMOD entry 346  
 usage notes  
   avoiding SYSMOD termination 337  
   deleted elements 338  
   ERROR indicator 338  
   exception SYSMOD data 338  
   IEANUC01 338  
   ineligible SYSMODs 336  
   restoring volumes 338  
   zone for SET BOUNDARY 333  
**restricting system utility programs 835**  
**restrictions**  
   CALLLIBS subentry 196, 698, 865  
   CIDTABL 865  
   EXRTYDD subentry 866  
   HFSCOPY subentry 866  
   LINK command and cross-zone information 201, 866  
   LIST subentry 866

**restrictions** *(continued)*

migrating from previous releases of SMP/E 865  
 MVS/370 and SMP/E 854  
**RETRY**  
   *See also* retry utility  
   ACCEPT command operand 15  
   APPLY command operand 63  
   excluding data sets 742  
   exit routine 801  
   including data sets 745  
   LINK command operand 203  
   OPTIONS entry 369, 744  
   RESTORE command operand 335  
   UTILITY entry for 787  
**retry utility**  
   default values 745, 787  
   defaults 744, 787, 836  
   OPTIONS entry 369, 744  
   UTILITY entry for 744, 787  
**RETRYDDN**  
   OPTIONS entry 369, 745  
**return codes**  
   RC operand 829  
   resetting 329  
   SMP/E commands 829  
   utilities  
     ACCEPT processing 22  
     APPLY processing 72  
     maximum acceptable value, specifying 790  
     summary 790  
**REUS**  
   LMOD entry 366, 701  
   MOD entry 368, 727  
**REUSE**  
   ACCEPT command operand 15  
   ACCEPT processing 45  
   APPLY command operand 64  
   APPLY processing 97  
**reusing assemblies**  
   ACCEPT processing 45  
   APPLY processing 97  
**REWORK**  
   ++APAR MCS operand 519  
   ++FUNCTION MCS operand 533  
   ++PTF MCS operand 574  
   ++USERMOD MCS operand 589  
   RECEIVE processing 252  
   SYSMOD entry  
     distribution zone 371, 765  
     global zone 776  
     target zone 371, 765  
**RFDSNPFX**  
   ++APAR MCS operand 520  
   ++FUNCTION MCS operand 534  
   ++PTF MCS operand 574

**RFDSNPF** (*continued*)

- ++USERMOD MCS operand 590
- conflict with DSPREFIX value 742

**RFPREFIX**

- RECEIVE command operand 241

**RLMOD**

- SYSMOD entry
  - distribution zone 765
  - target zone 765

**RMID**

- ++HFS MCS operand 538
- ++MAC MCS operand 555
- ++MOD MCS operand 566
- ++SRC MCS operand 583
- ACCEPT processing 38
- APPLY processing 87, 102
- data element entry 360, 649
- data element MCS operand 527
- HFS entry 365, 689
- MAC entry 367, 716
- MOD entry 368, 728
- SRC entry 370, 751
- updating at ACCEPT 47

**RMIDASM**

- MOD entry 368, 728

**RMODE=24**

- LMOD entry 366, 701
- MOD entry 368, 727

**RMODE=ANY**

- LMOD entry 366, 701
- MOD entry 368, 727

**RPL control blocks, dumping 143****S****S**

See SELECT

**SAMEDATASET**

- ZONERENAME command operand 435

**sample SMP/E cataloged procedure 813****sample USERMOD**

- for GIMMPDFT 824

**samples**

- MCS for
  - See data element MCS

**SAMPLIB members**

- MCS for
  - See data element MCS

**SAVEMTS**

- default value 745
- OPTIONS entry 369, 745

**SAVESTS**

- default value 745
- OPTIONS entry 369, 745

**SCDS**

- See also SMPSCDS
- CONVERT command operand 117

**SCRIPT files**

- MCS for
  - See data element MCS

**SCTR**

- LMOD entry 366, 701
- MOD entry 368, 727

**SELECT**

- ACCEPT command operand 16
- ACCEPT processing 30
- APPLY command operand 64
- APPLY processing 78
- RECEIVE command operand 241
- RECEIVE processing 242
- REJECT command operand 261
- REJECT processing 269
- RESTORE command operand 335
- RESTORE processing 341

**select-mode processing**

- ACCEPT command 30
- APPLY command 78
- LIST command 234
- RECEIVE command 251
- REJECT command 255, 269
- RESTORE command 337
- UNLOAD command 389

**service level of SMP/E 441****service routines**

- GIMDTS 803

**SET command**

- common errors 353
- data set sharing 354
- data sets required 350
- effect on dynamic allocation 350, 352
- ENQ considerations 354
- examples 350
- operands
  - BOUNDARY 349
  - OPTIONS 349
- processing 353
- summary 349
- syntax 349

**SETSSI statement**

- ++ZAP input 599

**SGGIMSMP (SMP/E SYSGEN macro), no longer supported 859****SHR**

- DDDEF entry 361, 657

**SHRPOOL, caution on use for LSR 867****single-CSI structure**

- converting 124

**skeletons for dialogs**

- MCS for
  - See data element MCS

**SMCCOR SOURCEID**

naming conventions 809

**SMCREC SOURCEID**

naming conventions 809

**SMP/E cataloged procedure 815****SMP/E control blocks, dumping 143****SMP/E data set entries 627****SMP/E data sets, descriptions of 607****SMP/E installation-wide exit routines***See* installation-wide exit routines for SMP/E**SMP/E MCS statements 517****SMP/E messages, tracing 143****SMP/E problems, debugging 143****SMP/E region size 821****SMP/E release summary 839****SMP/E reports 441****SMP/E return codes**

resetting 329

**SMP/E service level 441****SMP/E storage, dumping 143****SMP/E SYSGEN macro (SGGIMSMP), no longer supported 859****SMPCNTL, defining 609****SMPCSI***See also* CSI

converting 113

editing 405

HFS entry

distribution zone 687

target zone 687

listing 211

MAC entry

distribution zone 715

target zone 715

master CSI 815

MOD entry

distribution zone 723

target zone 723

OPTIONS entry 741

SRC entry

distribution zone 750

target zone 750

SYSMOD entry

distribution zone 759

global zone 774

target zone 759

TARGETZONE entry 782

unloading 379

updating with UCLIN 355

UTILITY entry 787

ZONESET entry 794

**SMPDEBUG**

DEBUG processing 148

defining 610

related to DUMPON operand for DEBUG 144

**SMPECNV SOURCEID**

naming conventions 138, 809

**SMPHOLD**

defining 610

RECEIVE processing 252, 253

**SMPJCLIN**

defining 610

used for JCLIN input 167

**SMPLIST**

defining 611

dynamically allocating with GIMMPDFT 825

**SMPLOG**

defining 611

listing 221, 237

user-written updates for 235

**SMPLOGA***See also* SMPLOG

defining 612

listing 221

**SMPLOTS**

defining 612

RESTORE processing 345

use for load modules 70

**SMPLOTS job, built by GENERATE for base version of load modules having a SYSLIB concatenation 163****SMPMTS**

ACCEPT processing 48, 739

APPLY processing 739

CLEANUP processing 107

defining 613

MTSMAC entry 369, 739

RESTORE processing 344

saving MTSMAC entries after ACCEPT processing (SAVEMTS) 745

scratching after system generation 171

updating with UCLIN 355

use as target macro library 70

**SMPnnnnn**

defining 621

**SMPOBJ**

defining 614

GENERATE processing 157

JCLIN processing 194

**SMPOUT**

default 827

defining 614

dynamically allocating with GIMMPDFT 825

**SMPPROC (cataloged procedure for SMP/E) 815****SMPPTFIN**

defining 614

RECEIVE processing 251, 252

**SMPPTS**

cleaning up (REJECT command) 255

defining 615

**SMPPTS** (*continued*)

- MCS entry 721
- RESTORE processing 347
- saving MCS entries after ACCEPT processing (NOPURGE) 743
- saving MCS entries after RESTORE processing (NOREJECT) 744
- sharing 834
- unexpected changes for (pending updates) 834

**SMPPUNCH**

- defining 615
- dynamically allocating with GIMMPDFT 825
- GENERATE processing 152, 162
- REPORT CALLLIBS processing 277
- REPORT CROSSZONE processing 290
- REPORT ERRSYSMODS processing 302
- REPORT SOURCEID processing 310
- REPORT SYSMODS processing 318
- UNLOAD processing 388

**SMRPT**

- defining 616
- dynamically allocating with GIMMPDFT 825

**SMPCDS**

- ACCEPT processing 48
- APPLY processing 85, 86, 91, 103
- BACKUP entries 359, 645
- CLEANUP processing 107
- converting 113
- defining 616
- listing 211
- RESTORE processing 338, 346
- updating with UCLIN 355

**SMPSNAP**

- DEBUG processing 148
- defining 617
- dynamically allocating with GIMMPDFT 825
- related to SNAP operand for DEBUG 144

**SMPSTS**

- ACCEPT processing 757
- APPLY processing 757
- CLEANUP processing 107
- defining 617
- RESTORE processing 344
- saving STSSRC entries after ACCEPT processing (SAVESTS) 745
- scratching after system generation 171
- STSSRC entry 370, 757
- updating with UCLIN 355
- use as target source library 70

**SMPTLIB**

- See also* relative files (RELFILES)
- ACCEPT processing 50, 743
- conflict between DSPREFIX value and RFDSNPFX value 742
- defaults
  - disposition 619

**SMPTLIB** (*continued*)

- defaults (*continued*)
  - space 742
- defining 618
- deleting
  - ACCEPT processing 50
  - REJECT processing 271
  - RESTORE processing 347
- DSPREFIX value for data set name
  - DDDEF entry 657
  - OPTIONS entry 742
- DSSPACE value for 742
- dynamically allocating
  - GIMMPDFT 827
  - RECEIVE command 250
- names 250
- RECEIVE processing 243
- REJECT processing 271
- RESTORE processing 347
- saving after ACCEPT processing (NOPURGE) 743

**SMPWRK1**

- defining 619
- dynamically allocating with GIMMPDFT 825

**SMPWRK2**

- defining 620
- dynamically allocating with GIMMPDFT 825

**SMPWRK3**

- defining 620
- dynamically allocating with GIMMPDFT 825

**SMPWRK4**

- defining 621
- dynamically allocating with GIMMPDFT 825

**SMPWRK6**

- defining 621
- dynamically allocating with GIMMPDFT 825

**SNAP**

- See also* SMPSNAP
- DEBUG command operand 144

**source**

- See also* SRC entry
- See also* STSSRC entry
- ++SRC MCS 582
- ++SRCUPD MCS 586
- adding 582
- assembling
  - ACCEPT processing 44
  - APPLY processing 95
- defining 582
- deleting
  - ++SRC MCS 582
  - ACCEPT processing 35
  - APPLY processing 83, 91
- moving 570
- replacing
  - ++SRC MCS 582
  - ACCEPT processing 43



**source** (*continued*)replacing (*continued*)

APPLY processing 94

updating

++SRCUPD MCS 586

ACCEPT processing 43

APPLY processing 94

**source elements for printers**

MCS for

*See* data element MCS**SOURCEID**

++ASSIGN MCS operand 521

ACCEPT command operand 16

ACCEPT processing 29

APPLY command operand 64

APPLY processing 78

assigning 242, 521

LIST command operand 225

naming conventions 808

PUTyynn 809

RECEIVE command operand 242

REJECT command operand 261

REJECT processing 268, 270

report 498

REPORT SOURCEID command operand 309

SMCCOR 809

SMCREC 809

SMPECNV 138, 809

SYSMOD entry

distribution zone 371, 765

global zone 374, 776

target zone 371, 765

UNLOAD command operand 385

**SPACE**

DDDEF entry 361, 659

**space problems***See* COMPRESS*See* PEMAX*See* RETRY*See* storage requirements for SMP/E**specifying zone to be updated 349****SRC***See also* source*See also* SRC entry

++MOVE MCS operand 570

CONVERT command operand 118

LIST command operand 225

MCS statement 582

SYSMOD entry

distribution zone 371, 765

global zone 777

target zone 371, 765

UNLOAD command operand 385

**SRC entry***See also* source**SRC entry** (*continued*)

++SRC MCS 582

created by JCLIN 184

listing 225, 751

subentries

DISTLIB 750

FMID 750

LASTUPD 750

LASTUPDTYPE 751

RMID 751

SYSLIB 751

UMID 751

summary 750

UCLIN for 370, 754

unloading 385, 753

**SRCUPD**

MCS statement 586

SYSMOD entry

distribution zone 371, 765

global zone 777

target zone 371, 765

**SREL**

++VER MCS operand 593

DLIBZONE entry 363, 676

GLOBALZONE entry 364, 682

RECEIVE processing 251

TARGETZONE entry 374, 782

**SSI**

++MAC MCS operand 555

++SRC MCS operand 583

**status report for SYSMODs 505****STD**

LMOD entry 366, 702

MOD entry 368, 727

**storage problems***See* COMPRESS*See* RETRY**storage requirements for SMP/E 821****STORCLAS**

DDDEF entry 659

**STORENX**

ACCEPT processing 46

APPLY processing 97

**STSSRC entry**

ACCEPT processing 48

RESTORE processing 344

saving through the OPTIONS entry 745

subentries 757

summary 757

UCLIN for 370, 757

**stub load module**

APPLY processing 84

**SUP***See also* SUPBY

++VER MCS operand 593

**SUP** (*continued*)

- LIST command operand 226
- UNLOAD command operand 385

**SUPBY**

- SYSMOD entry
  - distribution zone 371, 765
  - target zone 371, 765

**superseded SYSMODs**

- See also* LASTSUP
- See also* SUP
- See also* SUPBY
- See also* SUPING
- ACCEPT processing 32, 49
- APPLY processing 81, 104
- defining 593
- dummy entry for 49, 104

**superzap utility**

- ACCEPT processing 46
- APPLY processing 100
- default values 746, 787, 836
- examples 600
- OPTIONS entry 369, 746
- UTILITY entry for 746, 787

**SUPING**

- SYSMOD entry
  - distribution zone 371, 766
  - global zone 777
  - target zone 371, 766

**syntax of SMP/E commands**

- how to read 1
- rules for coding
  - commands 2
  - MCS statements 2
  - PARMLIB members 2

**SYS**

- See also* SYSTEM
- CONVERT command operand 118

**SYS1.HELP members**

- MCS for
  - See* data element MCS

**SYSALLDA**

- in sample USERMOD SMP0001 for module GIMMPDFT 826
- restriction for SMPTLIB data sets 244, 618

**SYSGEN**

- See* system generation

**SYSGEN macro**

- SMP/E (SGGIMSMP)
  - no longer supported 859

**SYSLIB**

- ++DELETE MCS operand 530
- ++HFS MCS operand 538
- ++MAC MCS operand 555
- ++MACUPD MCS operand 560
- ++MOVE MCS operand 571

**SYSLIB** (*continued*)

- ++SRC MCS operand 583
- ++SRCUPD MCS operand 586
- allocation for link-edits 99
- concatenation 818
- copied from DLIB entry 103, 670
- data element entry 360, 649
- data element MCS operand 527
- defining 622
- determining at APPLY 67
- DLIB entry 363, 671
- HFS entry 365, 689
- LMOD entry 366, 702
- MAC entry 716
- SRC entry 370, 751

**SYSLIB DD statement**

- See also* SYSLIB
- JCLIN processing 195
- link-edit steps 196
- PATH operand for HFS pathname 196

**SYSLMOD DD statement**

- JCLIN processing 197
- link-edit steps 197
- PATH operand for HFS pathname 197

**SYSMOD**

- See also* SYSMOD entry
- See also* SYSMOD ID
- CONVERT command operand 118
- LIST command operand 226
- RECEIVE command operand 242
- RECEIVE processing 242
- UNLOAD command operand 386

**SYSMOD Comparison report 500****SYSMOD entry**

- distribution zone
  - ACCEPT processing 49
  - summary 759
- global zone
  - ACCEPT processing 50
  - APPLY processing 106
  - saving after ACCEPT processing (NOPURGE) 743
  - saving after RESTORE (NOREJECT) 744
  - summary 774
- listing 226, 767, 778

**subentries**

- ACCDATE 762
- ACCEPT 759
- ACCID 774
- ACCTIME 762
- APAR 759, 774
- APPDATE 762
- APPID 775
- APPLY 759
- APPTIME 762
- ASSEM 759

**SYSMOD entry (continued)**

## subentries (continued)

BYPASS 760  
 CIFREQ 760  
 DELBY 760  
 DELETE 760, 775  
 DELLMOD 760  
 DLMOD 760  
 DSPREFIX 777  
 ELEMMOV 761  
 EMOVE 761  
 ERROR 761, 775  
 FESN 761, 775  
 FMID 761, 775  
 FUNCTION 761, 775  
 HFS 761, 775  
 IFREQ 761  
 INSTALLDATE 762  
 INSTALLTIME 762  
 JCLIN 762, 775  
 LASTSUP 762  
 LASTUPD 762  
 LASTUPDTYPE 762  
 MAC 763, 775  
 MACUPD 763, 776  
 MOD 763, 776  
 NPRES 763, 776  
 PRE 763, 776  
 PTF 763, 776  
 RECDATE 763, 776  
 RECTIME 763, 776  
 REGEN 764  
 RENLMOD 764  
 REQ 764, 776  
 RESTIME 764  
 RESTORE 764  
 REWORK 765, 776  
 RLMOD 765  
 SOURCEID 765, 776  
 SRC 765, 777  
 SRCUPD 765, 777  
 SUPBY 765  
 SUPING 766, 777  
 SZAP 766, 777  
 UCL syntax 759, 774  
 UCLDATE 766  
 UCLTIME 766  
 USERMOD 766, 777  
 VERNUM 766  
 VERSION 767, 777  
 XZAP 767, 777  
 summary 759, 774  
 target zone  
   ACCEPT processing 104  
   summary 759

**SYSMOD entry (continued)**

UCLIN for  
   distribution zone 371, 771  
   global zone 374, 780  
   target zone 371, 771  
 unloading 386, 770

**SYSMOD ID**

++APAR MCS operand 520  
 ++FUNCTION MCS operand 534  
 ++HOLD MCS operand 543  
 ++PTF MCS operand 575  
 ++RELEASE MCS operand 578  
 ++USERMOD MCS operand 590  
 naming conventions 809

**SYSMOD Regression report 503****SYSMOD Status report 505****SYSMODIDS**

REPORT SOURCEID command operand 309

**SYSMODS**

See SYSMOD entry

**SYSMODS**

REPORT SYSMODS command operand 318

**SYSOUT**

DDDEF entry 361, 659  
 dynamically allocating with GIMMPDFT 825

**SYSPRINT**

default 827  
 defining 622  
 dynamically allocating with GIMMPDFT 825  
 specified in UTILITY entry 789

**SYSPUNCH**

defining 623  
 GENERATE processing 157  
 JCLIN processing 194  
 special DISTLIB at ACCEPT 46

**SYSTEM**

See also SYSMOD ID  
 See also SYSTEM entry  
 ++HOLD MCS operand 541  
 ++RELEASE MCS operand 576  
 reason IDs 542, 577, 807

**SYSTEM entry**

conversion 140

**system generation**

compared to SMP/E GENERATE command 149  
 indicated by REGEN subentry 49, 764  
 related to JCLIN 172  
 used with GENERATE command 152, 165

**system libraries**

See SYSLIB

**system modification**

See SYSMOD

**system reason IDs**

ACCEPT command operand 9  
 ACCEPT processing 33

**system reason IDs** *(continued)*

APPLY command operand 56  
 APPLY processing 82  
 naming conventions 807  
 values  
   ACTION 9, 57, 542, 577, 807  
   AO 9, 57, 542, 577, 807  
   DELETE 9, 57, 542, 577, 807  
   DEP 9, 57, 542, 577, 807  
   DOC 9, 57, 542, 577, 807  
   EC 9, 57, 542, 577, 807  
   EXRF 9, 57, 542, 577, 808  
   FULLGEN 9, 57, 542, 577, 808  
   IOGEN 9, 57, 542, 577, 808  
   MSGSKEL 9, 57, 542, 577, 808  
   MVSCP 9, 57, 543, 578, 808

**system release**

See SREL

**system utility programs**

See utility programs

**SYSTSPRT**

specified in UTILITY entry 789

**SYSUDUMP**

dynamically allocating with GIMMPDFT 825

**SYSUT1**

defining 623  
 dynamically allocating with GIMMPDFT 825

**SYSUT2**

dynamically allocating with GIMMPDFT 825

**SYSUT3**

dynamically allocating with GIMMPDFT 825

**SYSUT4**

defining 624  
 dynamically allocating with GIMMPDFT 825

**SZAP**

SYSMOD entry  
 distribution zone 371, 766  
 global zone 777  
 target zone 371, 766

**T****tables for dialogs**

MCS for  
 See data element MCS

**TALIAS**

++MOD MCS operand 566  
 ++ZAP MCS operand 598  
 ACCEPT processing 20  
 APPLY processing 70  
 MOD entry 368, 728  
 RECEIVE processing 251

**target libraries**

compressing 91, 343  
 defining 624

**target libraries** *(continued)*

removing SYSMODs from (RESTORE  
 command) 333

**target zone**

ASSEM entry 640  
 cross-zone, automatic updating 783  
 cross-zone, information added by LINK  
 command 783  
 data element entry 648  
 DDDEF entry 654  
 DLIB entry 670  
 HFS entry 687  
 LMOD entry 698  
 MAC entry 715  
 MOD entry 723  
 sharing 833  
 SRC entry 750  
 SYSMOD entry 759  
 TARGETZONE entry 782  
 updating with JCLIN data 167  
 zone description provided by user 784

**TARGETZONE**

See also TARGETZONE entry  
 LIST command operand 227  
 REJECT command operand 262  
 REJECT processing 270  
 REPORT CROSSZONE command operand 288  
 ZONEDELETE command operand 400

**TARGETZONE entry**

listing 227, 784  
 subentries  
   name 782  
   OPTIONS 782  
   RELATED 782  
   SREL 782  
   TIEDTO 783  
   XZLINK 783  
   ZONEDESCRIPTION 784  
 summary 782  
 UCLIN for 374, 784  
 updating cross-zone subentries  
   ZONEEDIT command 405

**TEXT**

++HFS MCS operand 538  
 HFS entry 365, 687

**text files with SCRIPT tags**

MCS for  
 See data element MCS

**text library**

See TXLIB

**tgllib data set**

See target libraries

**TIEDTO**

TARGETZONE entry 783

**TLIB**

See SMPTLIB

**TLIBPREFIX**

SYSMOD entry  
global zone 777

**TODISTLIB**

++MOVE MCS operand 571

**TONAME**

++RENAME MCS operand 580

**TOSYSLIB**

++MOVE MCS operand 571  
TOSYSLIB 571

**totally copied library**

See also DLIB entry  
JCLIN processing 186, 670

**TOTYPE**

ZONERENAME command operand 435

**tracing SMP/E messages 143****TRACKS**

DDDEF entry 361, 654

**transformed elements**

ACCEPT processing 47  
APPLY processing 101

**TXLIB**

++HFS MCS operand 538  
++JCLIN MCS operand 549  
++MAC MCS operand 556  
++MOD MCS operand 566  
++SRC MCS operand 583  
data element MCS operand 527  
defining 624

**txlib data set**

See TXLIB

**TYPE operand**

PARMLIB statement 604

**TZONE**

See also TARGETZONE  
REPORT CROSSZONE processing 295

**U****UCL statements**

ADD 356  
DEL 356  
REP 356

**UCL syntax**

ASSEM entry 359, 640  
BACKUP entries 359  
data element entry 648  
distribution zone 360  
target zone 360  
DDDEF entry  
distribution zone 361, 654  
target zone 361, 654  
DLIB entry 363, 671

**UCL syntax (continued)**

DLIBZONE entry 363, 675  
FMIDSET entry 364, 679  
GLOBALZONE entry 364, 682  
HFS entry  
distribution zone 365  
summary 687  
target zone 365  
LMOD entry  
distribution zone 366  
summary 698  
target zone 366  
MAC entry  
distribution zone 367  
summary 715  
target zone 367  
MOD entry  
distribution zone 368  
summary 723  
target zone 368  
MTSMAC entry 369  
OPTIONS entry 369, 741  
SRC entry  
distribution zone 370  
summary 750  
target zone 370  
STSSRC entry 370  
SYSMOD entry  
distribution zone 371, 759  
global zone 374, 774  
target zone 371, 759  
TARGETZONE entry 374, 782  
UTILITY entry 375  
summary 788  
ZONESET entry 375, 794  
**UCLDATE**  
SYSMOD entry  
distribution zone 371, 766  
target zone 371, 766  
**UCLIN class value**  
ACCEPT command 8, 56, 541, 808  
**UCLIN command**  
alternative to (ZONEEDIT) 405  
data set sharing 378  
data sets required 375  
ENQ considerations 378  
examples 376  
operands  
RC 357  
output  
reports 375  
processing 377  
summary 355  
syntax 356  
UCL statements 357

**UCLIN command** (*continued*)  
usage notes 375  
zone for SET BOUNDARY 355

**UCLTIME**

SYSMOD entry  
distribution zone 371, 766  
target zone 371, 766

**UMID**

++MAC MCS operand 556  
++MOD MCS operand 567  
++SRC MCS operand 584  
ACCEPT processing 38, 48  
APPLY processing 87  
MAC entry 367, 717  
MOD entry 368, 728  
SRC entry 370, 751  
updating at APPLY 103

**unconditional requisites**

ACCEPT processing 32  
APPLY processing 80

**undefined data types**

MCS for  
See data element MCS

**unexpected changes (pending updates) 834**

**UNIT**

DDDEF entry 361, 660

**UNLOAD command**

data set sharing 389  
data sets required 388  
ENQ considerations 389  
examples 388  
modes of processing  
mass mode 389  
select mode 389  
operands  
EXSRCID 381  
XZLMODP 386  
XZMODP 387  
processing 388  
reports 388, 508  
SMPPUNCH output 388  
summary 379  
summary report 508  
syntax 379  
syntax notes 387  
zone for SET BOUNDARY 379

**UPDATE**

See also update utility  
++JCLIN MCS operand 549  
JCLIN command operand 169  
JCLIN processing 180, 198  
OPTIONS entry 369, 745  
UTILITY entry for 787

**update utility**

ACCEPT processing 42

**update utility** (*continued*)

APPLY processing 93  
default values 745, 787, 836  
JCLIN processing 198  
OPTIONS entry 369, 745  
restrictions 789  
specifying on JCLIN 169, 549  
UTILITY entry for 745, 787

**updating data set entries**

UCL statements 357

**updating SMPLOG 235**

**updating the target zone with JCLIN data 167**

**USER**

See also user reason IDs  
++HOLD MCS operand 541  
++RELEASE MCS operand 576  
reason IDs 543, 578, 808

**user exit routines**

See installation-wide exit routines for SMP/E

**user modification**

See USERMOD

**user reason IDs**

ACCEPT processing 34  
APPLY processing 82  
naming conventions 808

**user-defined data types**

MCS for  
See data element MCS

**user-written changes for SMPLOG 235**

**USERMOD**

See also ++USERMOD MCS  
ACCEPT command operand 17  
ACCEPT processing 29  
APPLY command operand 65  
APPLY processing 78  
defining 589  
LIST command operand 227  
naming conventions 810  
REJECT command operand 262  
REJECT processing 268, 269  
source update example 587  
SYSMOD entry  
distribution zone 371, 766  
global zone 777  
target zone 371, 766  
UNLOAD command operand 386

**USERMODs**

See ++USERMOD MCS

**UTILITY**

See also UTILITY entry  
See also utility programs  
LIST command operand 227

**UTILITY entry**

ACCEPT processing 46  
APPLY processing 98

**UTILITY entry** *(continued)*

GENERATE processing 160  
 listing 227, 791  
 subentries  
   LIST 788  
   NAME 788  
   PARM 788  
   PRINT 789  
   RC 790  
   summary 788  
 summary 787  
 UCLIN for 375, 791  
 updating multiple entries  
   ZONEEDIT command 405

**utility input**

MCS for  
   See data element MCS

**utility output**

MCS for  
   See data element MCS

**utility programs**

default programs  
   access method services (IDCAMS) 836  
   assembler (IEV90) 836  
   compress (IEBCOPY) 836  
   copy (IEBCOPY) 836  
   HFS copy (BPXCOPY) 836  
   link-edit utility (IEWBLINK) 836  
   link-edit utility (IEWL) 836  
   retry (IEBCOPY) 836  
   superzap (IMASPZAP) 836  
   update (IEBUPDTE) 836  
 default values  
   access method services (AMS) 741, 787  
   assembler 741, 787  
   compress 741, 787  
   copy 742, 787  
   HFS copy 743, 787  
   installation on VS1 systems 743, 787  
   link-edit utility 743, 787  
   retry 744, 787  
   superzap 746, 787  
   update 745, 787  
 loading 835  
 OPTIONS entry pointer to  
   access method services 741  
   assembler 741  
   compress 741  
   copy 742  
   HFS copy 743  
   IEHIOSUP utility 743  
   link-edit utility 743  
   retry 744  
   superzap 746  
   update 745

**utility programs** *(continued)*

parameters passed to 787, 788  
 restricting through GIMUTTBL 835  
 return codes for  
   ACCEPT processing 22  
   APPLY processing 72  
   default values 787  
   threshold 790  
 SYSPRINT output 787, 789  
 UTILITY entries for 787

**V****VER**

MCS statement 591

**VERNUM**

SYSMOD entry  
   distribution zone 371, 766  
   target zone 371, 766

**VERSION**

++HFS MCS operand 539  
 ++MAC MCS operand 556  
   ACCEPT processing 40  
   APPLY processing 89  
 ++MOD MCS operand 567  
   ACCEPT processing 40  
   APPLY processing 89  
 ++SRC MCS operand 584  
   ACCEPT processing 40  
   APPLY processing 89  
 ++VER MCS operand 593  
   ACCEPT processing 40  
   APPLY processing 89  
 data element MCS operand 527  
 SYSMOD entry  
   distribution zone 371, 767  
   global zone 777  
   target zone 371, 767

**VOLUME**

DDDEF entry 660

**VS1 installation utility**

default values 787

**VSAM problems 143****VSAM RPL control blocks, dumping 143****W****WAIT**

DDDEF entry 660  
 EXEC statement parameter for GIMSMP 814, 833

**WAITFORDSN**

DDDEF entry 361, 660

**X****x37 abends**

See COMPRESS

See PEMAX

See RETRY

See storage requirements for SMP/E

**XREF**

LIST command operand 228

sample LIST output

ASSEM entry 641

data element entry 650

HFS entry 690

LMOD entry 705

MAC entry 718

MOD entry 731

SRC entry 752

SYSMOD entry 770

**XZAP**

SYSMOD entry

distribution zone 371, 767

global zone 777

target zone 371, 767

**XZLINK**

TARGETZONE entry 783

**XZLMOD**

MOD entry 728

**XZLMODP**

LIST command operand 228

MOD entry 729

UNLOAD command operand 386

**XZMOD**

LMOD entry 703

**XZMODP**

LIST command operand 228

LMOD entry 703

UNLOAD command operand 387

**Z****ZAP**

See also superzap utility

MCS statement 598

OPTIONS entry 369, 746

UTILITY entry for 787

**ZCOPY**

See ZONECOPY command

**ZDEL**

See ZONEDELETE command

**ZDESC**

See ZONEDESCRIPTION

**ZEDIT**

See ZONEEDIT command

**ZEXP**

See ZONEEXPORT command

**ZIMP**

See ZONEIMPORT command

**ZMERGE**

See ZONEMERGE command

**ZONE**

ZONESET entry 375, 794

**zone sharing**

command phases 832

summary 831

types of zone access 831

**ZONECOPY command**

cross-zone subentries 393, 396

data set sharing 396

data sets required 392

ENQ considerations 396

examples 394

operands

INTO 391

OPTIONS 392

RC 392

RELATED 392

processing 396

reports 394

summary 391

syntax 391

usage notes 393

zone for SET BOUNDARY 391

**ZONEDELETE command**

cross-zone subentries 401, 402

data set sharing 402

data sets required 400

ENQ considerations 402

examples 401

operands

DLIBZONE 399

RC 399

TARGETZONE 400

output 401

processing 402

summary 399

syntax 399

usage notes 400

zone for SET BOUNDARY 399

**ZONEDESCRIPTION**

DLIBZONE entry 363, 676

GLOBALZONE entry 364, 683

TARGETZONE entry 374, 784

**ZONEEDIT command**

alternative to UCLIN 405

data set sharing 410

data sets required 408

ENQ considerations 410

examples 409

operands

CHANGE 407

entry type 406



**ZONEEDIT command** *(continued)*operands *(continued)*

- from value 407
- IF THEN 408
- RC 406
- subentry 407
- to value 407
- processing 410
- summary 405
- summary report 510
- syntax 405
- zone for SET BOUNDARY 405

**ZONEEXPORT command**

- cross-zone subentries 415, 416
- data set sharing 416
- data sets required 414
- ENQ considerations 416
- examples 415
- operands
  - INDEX 414
  - NOPURGE 414
  - OUTFILE 413
  - PURGE 414
  - RC 414
- processing 415
- summary 413
- syntax 413
- usage notes 415
- zone for SET BOUNDARY 413

**ZONEIMPORT command**

- cross-zone subentries 419, 421
- data set sharing 421
- data sets required 419
- ENQ considerations 421
- examples 419
- operands
  - INFILE 418
  - INTO 418
  - OPTIONS 418
  - RC 418
  - RELATED 418
- processing 420
- summary 417
- syntax 417
- usage notes 419
- zone for SET BOUNDARY 417

**ZONEINDEX**

- deleting with the ZONEEXPORT command 414
- GLOBALZONE entry 364, 683

**ZONEMERGE command**

- cross-zone subentries 425
- data set sharing 431
- data sets required 425
- ENQ considerations 431
- examples 426

**ZONEMERGE command** *(continued)*

## operands

- CONTENT 424
- DEFINITION 424
- INTO 424
- name 424
- NOREPLACE 424
- RC 424
- REPLACE 425
- processing 429
- summary 423
- summary report 512
- syntax 424
- usage notes 425
- zone for SET BOUNDARY 423

**ZONERENAME command**

- cross-zone subentries 439
- data set sharing 440
- data sets required 436
- ENQ considerations 440
- examples 437
- operands
  - NEWDATASET 434
  - old zone name 434
  - OPTIONS 434
  - RC 434
  - RELATED 435
  - SAMEDATASET 435
  - TO 434
  - TOTYPE 435
- processing 439
- summary 433
- syntax 434
- usage notes 436
- zone for SET BOUNDARY 433

**zones**

- copying
  - ZONECOPY command 391
  - ZONEEXPORT command 413
  - ZONEIMPORT command 417
  - ZONEMERGE command 423
- defining 624
- deleting
  - ZONEDELETE command 399
  - ZONEEXPORT command 414
- editing
  - ZONEEDIT command 405
- exporting
  - ZONEEXPORT command 413
- importing
  - ZONEIMPORT command 417
- merging
  - ZONEMERGE command 423
- naming conventions 675, 782
- renaming
  - ZONERENAME command 433

### **zones** (*continued*)

- sharing 831
- specifying on SET command 349

### **ZONES**

- REPORT CALLLIBS command operand 276
- REPORT ERRSYSMODS command operand 301
- REPORT SOURCEID command operand 310

### **ZONESET**

- See also* ZONESET entry
- LIST command operand 229
- REPORT CROSSZONE command operand 289

### **ZONESET entry**

- listing 229, 794
- subentries
  - ZONE 794
- summary 794
- UCLIN for 375, 795

### **ZREN**

- See* ZONERENAME command

1

2

3

4

5

---

# Communicating Your Comments to IBM

System Modification Program Extended  
Reference  
Release 8.1

Publication No. SC28-1107-13

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:  
FAX: (International Access Code)+1+914+385-0662
- If you prefer to send comments electronically, use this network ID:
  - IBMLink: KGNRCF at KGNVMC
  - IBM Mail Exchange: USIB27BQ at IBMMAIL
  - Internet: kgncrf@vnet.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

---

# Readers' Comments — We'd Like to Hear from You

## System Modification Program Extended

Reference

Release 8.1

Publication No. SC28-1107-13

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

**Note:** Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Today's date: \_\_\_\_\_

What is your occupation?

Newsletter number of latest Technical Newsletter (if any) concerning this publication:

How did you use this publication?

- |                          |                               |                          |                        |
|--------------------------|-------------------------------|--------------------------|------------------------|
| <input type="checkbox"/> | As an introduction            | <input type="checkbox"/> | As a text (student)    |
| <input type="checkbox"/> | As a reference manual         | <input type="checkbox"/> | As a text (instructor) |
| <input type="checkbox"/> | For another purpose (explain) |                          |                        |

---

---

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number:

Comment:

\_\_\_\_\_  
Name

\_\_\_\_\_  
Address

\_\_\_\_\_  
Company or Organization

\_\_\_\_\_  
Phone No.



Cut or Fold  
Along Line

Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES



**BUSINESS REPLY MAIL**

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation  
Department 52QA MS 911  
ENTERPRISE DRIVE  
KINGSTON NY 12401-1099



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold  
Along Line

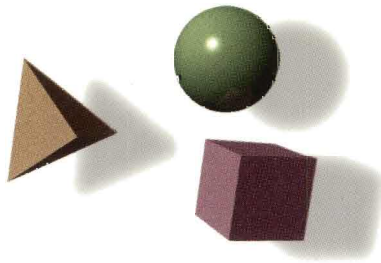




Program Number: 5668-949



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber



SC28-1107-13

