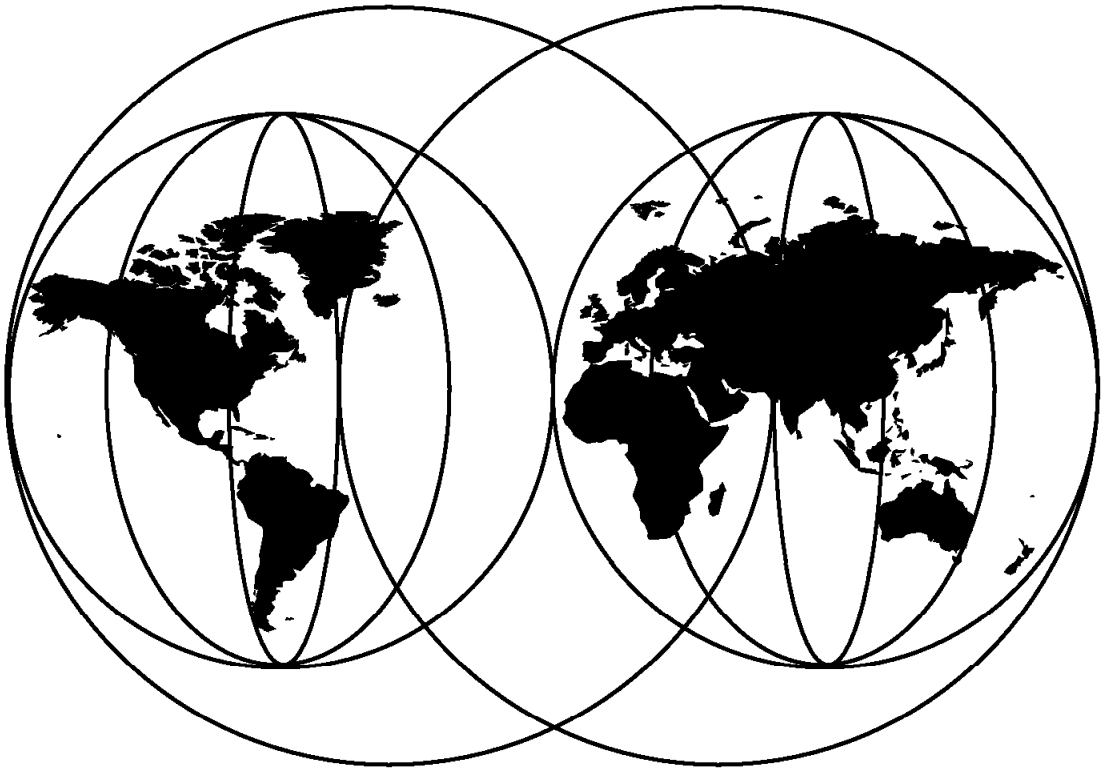
ABCs of OS/390 System Programming Volume 3

P. Rogers, G. Capobianco, D. Carey, N. Davies, L. Fadel, K. Hewitt, J. Oliveira, F. Pita, A. Salla, V. Sokal, Y. F. Tay, H. Timm



International Technical Support Organization

www.redbooks.ibm.com

SG24-5653-00

International Technical Support Organization

SG24-5653-00

ABCs of OS/390 System Programming Volume 3

April 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix A, “Special Notices” on page 331.

First Edition (April 2000)

This edition applies to OS/390 Version 2 Release 8, Program Number 5647-A01, and to all subsequent releases and modifications.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, New York 12601-5400

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

Contents

Figures ix

Tables xv

Preface xvii
The team that wrote this redbook xvii
Comments welcome xviii

Chapter 1. Introduction to DFSMS/MVS 1
1.1 Introduction to data management 2
1.2 Data sets 4
1.2.1 Data set name rules 5
1.2.2 Logical records 6
1.2.3 Record formats 6
1.2.4 Data set organization (DSORG) 7
1.2.5 Locating a data set 9
1.2.6 Cataloged and uncataloged data sets 11
1.3 Volume Table of Contents (VTOC) 12
1.3.1 Data set control block (DSCB) types 13
1.3.2 Index VTOC structure 15
1.3.3 Creating the VTOC and index VTOC 15
1.4 ICKDSF 16
1.4.1 Initializing a DASD volume 16
1.4.2 VTOC and index VTOC 17
1.4.3 ICKDSF stand-alone version 17
1.4.4 Problem determination 19
1.5 Traditional DASD 21
1.6 Redundant Array of Independent Disks (RAID) 22
1.7 RVA highlights 24
1.8 Seascape architecture 26
1.8.1 Powerful storage server 26
1.8.2 Enterprise Storage Server 29
1.8.3 Serial Storage Architecture (SSA) 31
1.8.4 ESS universal access 33
1.8.5 Operating systems supporting ESS 34
1.8.6 ESS new performance functions 35
1.8.7 WLM controlling PAVs 37
1.8.8 ESS copy services 38
1.8.9 StorWatch product highlights 40
1.9 Introduction to tape processing 41
1.9.1 Describing the labels 43
1.9.2 Initializing tape cartridges 45
1.9.3 Tape capacity 47
1.9.4 3494 tape library 49
1.10 Introduction to VTS 51
1.11 Introduction to utilities 53
1.11.1 System utilities programs 53
1.11.2 Data set utility programs 54
1.11.3 IEFBR14 55
1.11.4 IEBCOMPR (compare data set) program 56
1.11.5 Data sets that can be compared 58

1.11.6 Data sets that cannot be compared . . . 59
1.12 IEBCOPY . . . 60
1.12.1 IEBCOPY copy operation . . . 62
1.12.2 IEBCOPY compress operation . . . 64
1.13 IEBGENER . . . 65
1.13.1 Adding members to a PDS using IEBGENER . . . 66
1.13.2 Copying data to tape . . . 67
1.14 IEHLIST . . . 68
1.14.1 LISTVTOC output . . . 69
1.15 Access method services . . . 70
1.15.1 Invoking access method services . . . 71
1.15.2 Functional commands . . . 72
1.16 Data Collection Facility (DCOLLECT) . . . 74
1.16.1 AMS modal commands . . . 76
1.17 Generation data groups (GDG) . . . 77
1.17.1 Defining a generation data group . . . 79
1.17.2 Absolute generation and version numbers . . . 81
1.17.3 Relative generation number . . . 83
1.17.4 Rolled in and rolled off . . . 84
1.18 Access method functions . . . 85
1.18.1 Major DFSMS/MVS access methods . . . 87
1.18.2 BPAM to access PDS and PDSE . . . 88
1.18.3 PDS and PDSE data organizations . . . 89
1.18.4 PDSE structure . . . 91
1.18.5 Sequential access methods . . . 92
1.19 Virtual Storage Access Method (VSAM) . . . 94
1.19.1 VSAM resource pool . . . 95
1.19.2 VSAM components . . . 97
1.19.3 Key sequenced data set (KSDS) . . . 99
1.19.4 Data/Index relationship . . . 100
1.19.5 Relative record data set (RRDS) . . . 102
1.19.6 Typical RRDS processing . . . 103
1.19.7 Linear data set (LDS) . . . 104
1.19.8 Data-in-virtual . . . 105
1.19.9 Data-in-virtual objects . . . 106
1.19.10 Mapping a linear data set . . . 107
1.19.11 Entry sequenced data set (ESDS) . . . 108
1.19.12 Typical ESDS processing (ESDS) . . . 109
1.20 DFSORT . . . 110
1.21 DFSMS/MVS Network File System . . . 112
1.22 DFSMS/MVS Optimizer . . . 114
1.23 DFSMSdss . . . 115
1.23.1 DFSMSdss: physical and logical processing . . . 116
1.23.2 DFSMSdss: logical processing . . . 117
1.23.3 DFSMSdss: physical processing . . . 119
1.23.4 DFSMSdss stand-alone services . . . 121
1.24 DFSMShsm . . . 122
1.24.1 Availability management . . . 123
1.24.2 Space management . . . 125
1.24.3 Storage device hierarchy . . . 127
1.24.4 HSM volume types . . . 128
1.24.5 Automatic space management . . . 130
1.24.6 Recall . . . 131
1.25 Removable media manager (DFSMSrmm) . . . 133
1.25.1 Libraries and locations . . . 134

1.26 What DFSMSrmm can manage . . . 135
1.26.1 Removable media library . . . 135
1.26.2 Storage location . . . 136
1.26.3 Managing libraries and storage locations . . . 137

Chapter 2. Storage management . . . 139
2.1 DFSMS/MVS environment . . . 140
2.1.1 The DFSMS/MVS functional components . . . 141
2.2 Introduction to system-managed storage (SMS) . . . 142
2.3 Benefits of system-managed storage . . . 143
2.4 Implementing your storage management policies . . . 146
2.5 Implementing and monitoring storage management policies . . . 148
2.5.1 Monitoring your policies . . . 149
2.6 Managing data with SMS . . . 150
2.6.1 How to be system-managed . . . 151
2.6.2 Using data classes . . . 153
2.6.3 Using storage classes . . . 155
2.6.4 Using management classes . . . 157
2.6.5 Management class functions . . . 159
2.6.6 Using storage groups . . . 160
2.6.7 Using aggregate backup and recovery support (ABARS) . . . 162
2.6.8 Using automatic class selection routines . . . 164
2.7 Defining the storage management subsystem configuration . . . 166
2.8 Activating a minimal SMS configuration . . . 168
2.8.1 Managing data with a minimal SMS configuration . . . 170
2.8.2 Steps for a minimal SMS configuration . . . 172
2.8.3 Allocating SMS control data sets . . . 173
2.8.4 Define GRS resource names for active SMS control data sets . . . 174
2.8.5 Defining a minimal SMS configuration . . . 175
2.8.6 DFSMS setup for OS/390 . . . 178
2.8.7 Activating and starting SMS . . . 180
2.8.8 Display SMS configuration . . . 182
2.8.9 Controlling SMS processing with MVS operator commands . . . 183
2.8.10 Enforcing standards with DC ACS routine . . . 185
2.9 Establishing installation standards . . . 186
2.9.1 Data types that can be system managed . . . 187
2.9.2 Data types that cannot be system managed . . . 189
2.9.3 Developing naming conventions . . . 191
2.9.4 Lowest-level qualifiers (LLQ) standards . . . 193
2.9.5 Simplifying JCL . . . 195
2.9.6 Allocating data . . . 196
2.9.7 Creating a VSAM cluster . . . 198
2.9.8 Using retention period and expiration date . . . 200
2.10 Managing data allocation . . . 202
2.10.1 Using data class to standardize data allocation . . . 202
2.10.2 Data class attributes . . . 204
2.10.3 Planning and defining data classes . . . 205
2.10.4 Ensuring device independence . . . 206
2.11 SMS PDSE support . . . 207
2.11.1 PDSE conversion . . . 209
2.11.2 Program objects . . . 211
2.11.3 Selecting data sets to allocate as PDSEs . . . 214
2.11.4 Allocating new PDSEs . . . 215
2.11.5 Identifying PDSEs . . . 216
2.12 Introduction to ISMF . . . 217

2.12.1 ISMF products relationship . . . 218
2.12.2 What you can do with ISMF . . . 220
2.12.3 Accessing ISMF . . . 222
2.12.4 Navigating through ISMF . . . 223
2.12.5 Selecting an option from the ISMF Primary Option menu . . . 225
2.12.6 ISMF Profile Option Menu . . . 226
2.12.7 Data Set Selection Entry Panel . . . 227
2.12.8 Data Set List panel . . . 228
2.12.9 Volume List Selection Menu . . . 229
2.12.10 ISMF Volume List panel . . . 230
2.12.11 Management Class Application Selection . . . 231
2.12.12 ISMF management class list . . . 232
2.12.13 Data Class Application Selection . . . 233
2.12.14 ISMF data class list . . . 234
2.12.15 Storage Class Application Selection . . . 235
2.12.16 ISMF storage class List . . . 236
2.12.17 Saved ISMF Lists . . . 237
2.13 Removable Media Manager (DFSMSrmm) . . . 238

Chapter 3. System Modification Program/Enhanced (SMP/E) . . . 239
3.1 Introduction to SMP/E . . . 240
3.2 SYSMODs . . . 241
3.2.1 Introducing an element - the function SYSMOD . . . 243
3.2.2 Preventing problems with an element (PTF) . . . 245
3.2.3 Fixing problems with an element - the APAR SYSMOD . . . 251
3.2.4 Customizing an element - the USERMOD SYSMOD . . . 253
3.3 Data sets used by SMP/E . . . 255
3.3.1 Dynamic allocation of SMP/E data sets . . . 258
3.3.2 Standard defaults . . . 260
3.3.3 How dynamic allocation works . . . 261
3.4 Consolidated Software Inventory (CSI) . . . 263
3.4.1 The organization of the CSI data set . . . 263
3.4.2 How to organize CSI data sets . . . 265
3.4.3 How to allocate a CSI data set . . . 266
3.4.4 How to initialize a CSI data set . . . 266
3.4.5 Defining zones for your system . . . 268
3.5 SMP/E commands you need to know . . . 270
3.5.2 Displaying SMP/E data . . . 271
3.6 Receiving SYSMODs . . . 272
3.6.1 Packaging of the SYSMODs . . . 272
3.6.2 The RECEIVE Process . . . 274
3.6.3 Managing exception SYSMOD through HOLLDATA . . . 275
3.6.4 SMP/E data sets used in the RECEIVE Process . . . 277
3.6.5 Reports for RECEIVE processing . . . 281
3.7 Rejecting SYSMODs . . . 284
3.7.1 Processing modes of the REJECT command . . . 284
3.8 The APPLY Process . . . 290
3.8.1 Selecting SYSMODS . . . 290
3.8.2 How SMP/E keeps track of APPLY processing . . . 291
3.8.3 The APPLY CHECK Process . . . 295
3.9 The RESTORE process . . . 297
3.9.1 Removing SYSMODs . . . 297
3.9.2 Selecting elements . . . 298
3.9.3 Replacing the elements in the target libraries . . . 298
3.9.4 How SMP/E keeps track of RESTORE processing . . . 298

3.9.5 The RESTORE command . . . 298
3.9.6 Restore examples . . . 299
3.10 The ACCEPT process . . . 302
3.10.1 Selecting SYSMODs . . . 302
3.10.2 Selecting elements . . . 303
3.10.3 Updating the distribution libraries . . . 303
3.10.4 How SMP/E keeps track of ACCEPT processing . . . 303
3.10.5 The ACCEPT command . . . 303
3.10.6 ACCEPT CHECK facility . . . 304
3.10.7 The reporting output . . . 304
3.10.8 ACCEPT examples . . . 305
3.11 Other useful SMP/E commands . . . 307
3.11.1 Using the LIST command . . . 309
3.11.2 Using the REPORT ERRSYSMOD command . . . 311
3.12 SMP/E dialogs . . . 316
3.12.1 Query Selection Menu . . . 317
3.12.2 CSI Query Panel . . . 318
3.12.3 CSI Query - Select Entry Panel . . . 319
3.12.4 CSI Query - SYSMOD Entry Panel . . . 320
3.12.5 Building SMP/E jobs by using dialog . . . 321
3.12.6 Command Generation Selection Menu . . . 322
3.12.7 Command Generation Selection Zone . . . 323
3.12.8 Command Generation - LIST Command . . . 324
3.12.9 List Global zone SYSMOD options . . . 325
3.12.10 Command Generation - list FORFMID . . . 326
3.12.11 Command Generation Selection Menu . . . 327
3.12.12 Command Generation - SUBMIT . . . 328
3.12.13 The generated job . . . 329

Appendix A. Special Notices . . . 331

Appendix B. Related Publications . . . 335
B.1 IBM Redbooks . . . 335
B.2 IBM Redbooks collections . . . 336
B.3 Other resources . . . 336

How to get IBM Redbooks . . . 339
IBM Redbooks fax order form . . . 340

Glossary . . . 341

IBM Redbooks evaluation . . . 355

Figures

1. Introduction to data management . . . 2
2. Data sets . . . 4
3. Data set name rules . . . 5
4. Logical record length . . . 6
5. PO data set . . . 7
6. Locating a data set . . . 9
7. Cataloged and uncataloged data sets . . . 11
8. Volume table of contents (VTOC) . . . 12
9. Data set control block (DSCB) . . . 13
10. Index VTOC structure . . . 15
11. Initializing a volume . . . 16
12. Problem determination . . . 19
13. Traditional DASD . . . 21
14. Redundant array of independent disks (RAID) . . . 22
15. RVA highlights . . . 24
16. Seascape architecture . . . 26
17. Enterprise Storage Server . . . 29
18. Serial Storage Architecture (SSA) . . . 31
19. ESS universal access . . . 33
20. Operating systems supporting ESS . . . 34
21. ESS new performance functions . . . 35
22. WLM controlling PAVs . . . 37
23. ESS copy services . . . 38
24. StorWatch product highlights . . . 40
25. Introduction to tape processing . . . 41
26. SL and NL . . . 43
27. Initializing tape cartridges . . . 45
28. IEHINITT example to write EBCDIC labels in different densities . . . 46
29. Place serial number on eight tape volumes . . . 46
30. Tape capacity . . . 47
31. 3494 tape library . . . 49
32. Introduction to VTS . . . 51
33. Utilities . . . 53
34. IEFBR14 . . . 55
35. IEBCOMPR . . . 56
36. Directories of PO that can be compared . . . 58
37. Directories of PO that cannot be compared . . . 59
38. IEBCOPY . . . 60
39. IEBCOPY copy operation . . . 62
40. IEBCOPY compress operation . . . 64
41. IEBGENER . . . 65
42. Adding members to a PDS using IEBGENER . . . 66
43. Copying data to tape . . . 67
44. IEHLIST . . . 68
45. LISTVTOC output . . . 69
46. Access method services . . . 70
47. Functional commands . . . 72
48. Data Collection Facility (DCOLLECT) . . . 74
49. AMS modal commands . . . 76
50. Generation data groups . . . 77
51. Defining a GDG . . . 79

52. Absolute generation and version numbers . . . 81
53. Relative generation numbers . . . 83
54. Access method functions . . . 85
55. Major DFSMS/MVS access methods . . . 87
56. BPAM and PDSE . . . 88
57. PDS and PDSE data organization . . . 89
58. PDSE structure . . . 91
59. Sequential access methods . . . 92
60. Virtual Storage Access Method . . . 94
61. VSAM resource pool . . . 95
62. VSAM components . . . 97
63. Key sequenced data set (KSDS) . . . 99
64. Data/Index relationship . . . 100
65. Relative record data set (RRDS) . . . 102
66. Typical RRDS processing . . . 103
67. Linear data set (LDS) . . . 104
68. Data-in-virtual . . . 105
69. Data-in-virtual objects . . . 106
70. Mapping a linear data set . . . 107
71. Entry sequenced data set (ESDS) . . . 108
72. Typical ESDS processing (ESDS) . . . 109
73. DFSORT . . . 110
74. DFSMS/MVS Network File System . . . 112
75. DFSMS/MVS Optimizer . . . 114
76. DFSMSdss . . . 115
77. DFSMSdss: physical and logical processing . . . 116
78. DFSMSdss: logical processing . . . 117
79. DFSMSdss: physical processing . . . 119
80. Stand-alone services . . . 121
81. Introduction to DFSMShsm . . . 122
82. Availability management . . . 123
83. Space management . . . 125
84. Storage device hierarchy . . . 127
85. HSM volume types . . . 128
86. Automatic space management . . . 130
87. Recall . . . 131
88. Introduction to DFSMSrmm . . . 133
89. Libraries and locations . . . 134
90. DFSMSrmm can manage . . . 135
91. Managing libraries and storage locations . . . 137
92. DFSMS/MVS environment . . . 140
93. DFSMS/MVS functional components . . . 141
94. System-managed storage environment . . . 142
95. Benefits of system-managed storage . . . 143
96. Implementing your storage management policies . . . 146
97. Implementing monitoring storage policies . . . 148
98. Monitoring your policies . . . 149
99. Managing data with SMS . . . 150
100. How to be system-managed . . . 151
101. Using data classes . . . 153
102. Using storage classes . . . 155
103. Using management class . . . 157
104. Management class functions . . . 159
105. Using storages groups . . . 160
106. Aggregate backup and recovery support . . . 162

107. Using automatic class selection . . . 164
108. Defining the storage management subsystem. . . . 166
109. Activating a minimal SMS configuration . . . 168
110. Managing data with a minimal SMS configuration . . . 170
111. Steps for a minimal SMS configuration . . . 172
112. Allocating SMS control data sets . . . 173
113. Defining a minimal SMS configuration . . . 175
114. DFSMS setup for OS/390 . . . 178
115. Activating and starting SMS . . . 180
116. Display SMS configuration . . . 182
117. Controlling SMS processing with commands . . . 183
118. Enforcing standards with DC ACS routine . . . 185
119. Establishing installation standards . . . 186
120. Data types that can be system managed . . . 187
121. Data types that cannot be system managed . . . 189
122. Highest-level qualifiers . . . 191
123. Lowest-level qualifiers (LLQ) standards . . . 193
124. Simplifying JCL . . . 195
125. Allocating data . . . 196
126. Creating a VSAM cluster . . . 198
127. Space parameter in a KSDS VSAM cluster . . . 199
128. Using retention period and expiration date . . . 200
129. Managing data allocation . . . 202
130. Data class attributes . . . 204
131. Planning and defining data classes . . . 205
132. Ensuring device independence . . . 206
133. SMS PDSE support . . . 207
134. PDSE conversion . . . 209
135. Program objects . . . 211
136. Selecting data sets to allocate as PDSEs . . . 214
137. Allocating a new PDSE . . . 215
138. Identifying a PDSE . . . 216
139. Introduction to ISMF . . . 217
140. ISMF products relationship . . . 218
141. What you can do with ISMF . . . 220
142. Accessing ISMF . . . 222
143. Navigating through ISMF . . . 223
144. ISMF Primary Option Menu . . . 225
145. ISMF Profile Option Menu . . . 226
146. Data Set Selection Entry Panel . . . 227
147. Data Set List Panel . . . 228
148. Volume List Selection Menu . . . 229
149. ISMF Volume List panel. . . . 230
150. Management Class Application Selection . . . 231
151. ISMF management class list . . . 232
152. Data Class Application Selection . . . 233
153. ISMF data class list . . . 234
154. Storage Class Application Selection . . . 235
155. ISMF Storage Class List . . . 236
156. Saved ISMF Lists . . . 237
157. Removable Media Manager (DFSMSrmm) . . . 238
158. SMP/E overview . . . 240
159. SYSMODs . . . 241
160. Introducing an element . . . 243
161. Example SYSMOD with four elements . . . 244

162. Preventing problems with an element (PTF) . . . 245
163. Example simple PTF SYSMOD . . . 246
164. PTF replacement . . . 247
165. PTF prerequisite . . . 248
166. Load module construction . . . 249
167. Fixing problems with an element (APAR) . . . 251
168. Example simple APAR SYSMOD . . . 251
169. Customizing an element - the USERMOD SYSMOD . . . 253
170. Example simple USERMOD SYSMOD . . . 253
171. SMP/E data sets . . . 255
172. Dynamic allocation . . . 258
173. Dynamic allocation check sequence . . . 261
174. Consolidated Software Inventory (CSI) . . . 263
175. Basic structure of CSI . . . 265
176. Sample JCL for allocating a CSI data set . . . 266
177. Sample job to initialize the CSI data set . . . 267
178. Relationship of zones in CSI . . . 268
179. Basic SMP/E commands . . . 270
180. The RECEIVE process . . . 272
181. The RECEIVE process . . . 274
182. Example SMPTLIB data sets . . . 275
183. Sources of HOLDDATA . . . 276
184. RECEIVE examples . . . 278
185. Sample JCL to process only HOLDDATA . . . 279
186. Sample JCL to process only the SYSMODs . . . 279
187. Sample JCL to receive both the SYSMODs and HOLDDATA . . . 280
188. Sample JCL to receive selected SYSMODs and HOLDDATA . . . 280
189. Reports for RECEIVE processing . . . 281
190. Sample RECEIVE Summary Report . . . 282
191. Sample RECEIVE Exception Data Report . . . 282
192. Sample RECEIVE File Allocation Report . . . 283
193. Sample RECEIVE Product Summary Report . . . 283
194. The REJECT process . . . 284
195. REJECT examples . . . 286
196. Sample JCL for rejecting SYSMODs in MASS mode . . . 287
197. Rejecting PTFs that have been received but not applied . . . 287
198. Rejecting a SYSMOD in SELECT mode . . . 287
199. Rejecting HOLDDATA from the system . . . 288
200. Rejecting in PURGE mode . . . 288
201. Rejecting in NOFMID mode . . . 288
202. The APPLY process . . . 290
203. APPLY examples . . . 293
204. Sample JCL to apply all SYSMODs from a given source . . . 294
205. Applying with the GROUP operand . . . 294
206. Applying with the CHECK operand . . . 294
207. The APPLY CHECK process . . . 295
208. The RESTORE process . . . 297
209. RESTORE examples . . . 299
210. Removing a single PTF . . . 299
211. Cleaning up the SMP records after the reject . . . 300
212. Restoring and reapplying PTFs . . . 300
213. Accepting some PTFs and then restoring another . . . 300
214. Restoring PTFs using the Group operand . . . 301
215. The ACCEPT process . . . 302
216. ACCEPT examples . . . 305

217. Accepting all SYSMODs from a given source 305
218. Accepting SYSMODs with ACCEPT 306
219. Using the GROUPEXTEND operand 306
220. Other SMP/E commands 307
221. The LIST command 309
222. Listing entries in a particular zone 310
223. Listing all DDDEF entries in the global zone and all defined zones . . 310
224. Listing all PTFs for a specific SYSMOD 310
225. The REPORT ERRSYSMODS command 311
226. Checking if HOLDDATA affects any already applied SYSMODs 312
227. Checking the effect of HOLDDATA on a specific SYSMOD 312
228. Reports for REPORT ERRSYSMODS 313
229. Sample Exception SYSMOD Report 314
230. Sample SMPPUNCH output 315
231. SMP/E Primary Option Menu 316
232. Query Selection Menu 317
233. CSI Query Panel 318
234. CSI Query - Select Entry Panel 319
235. CSI Query - SYSMOD Entry Panel 320
236. Building SMP/E jobs by using dialog 321
237. Command Generation Selection Menu 322
238. Command Generation Selection Zone 323
239. Command Generation - LIST Command 324
240. List Global zone SYSMOD options 325
241. Command Generation - list FORFMID 326
242. Command Generation Selection Menu 327
243. Command Generation - SUBMIT 328
244. The generated job 329

Tables

1. DSCBs that can be found in the VTOC 13
2. DASD capacity 21
3. Types of labels 43
4. Tape capacity of various IBM products 47
5. System utility programs 53
6. Data set utility programs 54
7. Functional commands 72

Preface

This redbook is Volume 3 of a five-volume set that is designed to introduce the structure of an OS/390 and S/390 operating environment. The set will help you install, tailor, and configure an OS/390 operating system, and is intended for system programmers who are new to an OS/390 environment.

In this Volume, Chapter 1 provides an introduction to the DFSMS/MVS. DFSMS/MVS is the data management part of the operating system that organizes, identifies, stores, catalogs, and retrieves all the data information used by an installation.

Chapter 2 provides an overview of DFSMS/MVS and its functional components.

Chapter 3 describes OS/390 SMP/E which is a tool designed to manage the installation of software products on your OS/390 system and to track the modifications you make to those products.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Poughkeepsie Center.

Paul Rogers is an OS/390 specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on various aspects of OS/390. Before joining the ITSO 11 years ago, he worked in the IBM Installation Support Center (ISC) in Greenford, England as OS/390 and JES support for IBM EMEA.

Guillermo Capobianco is an IT Specialist in IBM Global Services PSS Argentina. He has five years of experience working with customers on MVS, MVS-related program products, and OS/390. He is currently leading a technical group providing on-site customer support for the OS/390 platform.

David Carey is a Senior IT Availability Specialist with the IBM Support Center in Sydney, Australia, where he provides defect and nondefect support for CICS, CICSPlex/SM, MQSeries, and OS/390. David has 19 years of experience within the information technology industry, and was an MVS systems programmer for 12 years prior to joining IBM.

T. Nigel Davies is a Systems Specialist in IBM Global Services Product Support Services (PSS) in the United Kingdom. He has 10 years of IT experience in various roles, ranging from operations to PC and LAN support to mainframe systems programming. He joined IBM in 1997 with eight years of experience as a VM/VSE systems programmer, and since joining IBM has cross-trained in OS/390 systems skills. His areas of expertise include VM and VSE systems programming, installation, and technical support, and more recently, OS/390 installation and support. **Luiz Fadel**

Ken Hewitt is an IT Specialist in IBM Australia. He has over 10 years of experience working with S/390 customers in a range of roles from CE to System Engineer. His areas of expertise include I/O and OSA configuration.

Joao Natalino Oliveira

Joao Natalino de Oliveira is a certified I/T consulting specialist working for the S/390 in Brazil providing support for Brazil and Latin America. He has 24 years of experience in large systems including MVS-OS/390. His areas of expertise include performance and capacity planning, server consolidation and system programming. He has a bachelor degree in Math and Data Processing from Fundação Santo André Brazil.

Fabio Chaves Pita

Alvaro Salla has 30 years of experience in OS operating systems (since MVT). He has written several redbooks on S/390 subjects. Retired from IBM Brasil, he is now a consultant for IBM customers.

Valeria Sokal is an MVS system programmer at Banco do Brasil. She has 11 years of experience in the mainframe arena. Her areas of expertise include MVS, TSO/ISPF, SLR, and WLM.

Yoon Foh Tay is an IT Specialist with IBM Singapore PSS (S/390). He has six years of experience on the S/390 platform, providing on-site support to customers.

Hans-Juergen Timm is an Advisory Systems Engineer in IBM Global Services PSS Germany. He has 20 years of experience working with customers in the areas of MVS and OS/390, software and technical support, and planning and management. He also worked six years as an MVS Instructor in the IBM Education Centers in Mainz and Essen, Germany. His areas of expertise include implementation support for OS/390, Parallel Sysplex, UNIX System Services, and Batch Management.

Comments welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks evaluation” on page 355 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. Introduction to DFSMS/MVS

DFSMS/MVS has hardware requirements that include at least one processor, printer, and console, in addition to DASD and tape devices for storage.

The following licensed programs are prerequisites for installing and maintaining DFSMS/MVS:

- A base OS/390 system with the binder provided by DFSMS/MVS, or a linkage editor provided by MVS/DFP Version 3 (5665-XA3)
- Assembler H (5668-962), with current maintenance, or High Level Assembler for MVS (5696-234), with current maintenance
- System Modification Program Extended (SMP/E) (5668-946)

DFSMS/MVS also requires subsystem functions provided by a compatible level of either JES2 or JES3. *OS/390 Planning for Installation*, GC28-1726, documents the supported releases of JES2 and JES3 and also provides a list of the licensed programs required to install the OS/390 product.

The MVS operating system requires storage for DFSMS/MVS, for JES2 or JES3, for all licensed programs installed on the system, and for the various distribution and target libraries supported by each product. Storage requirements vary depending upon the products installed on your system and the DASD you choose to store on. Information on storage requirements is provided by the program directories or installation guides for these products.

Virtual storage and DASD space requirements for the installation of DFSMS/MVS are specified in the DFSMS/MVS Program Directory.

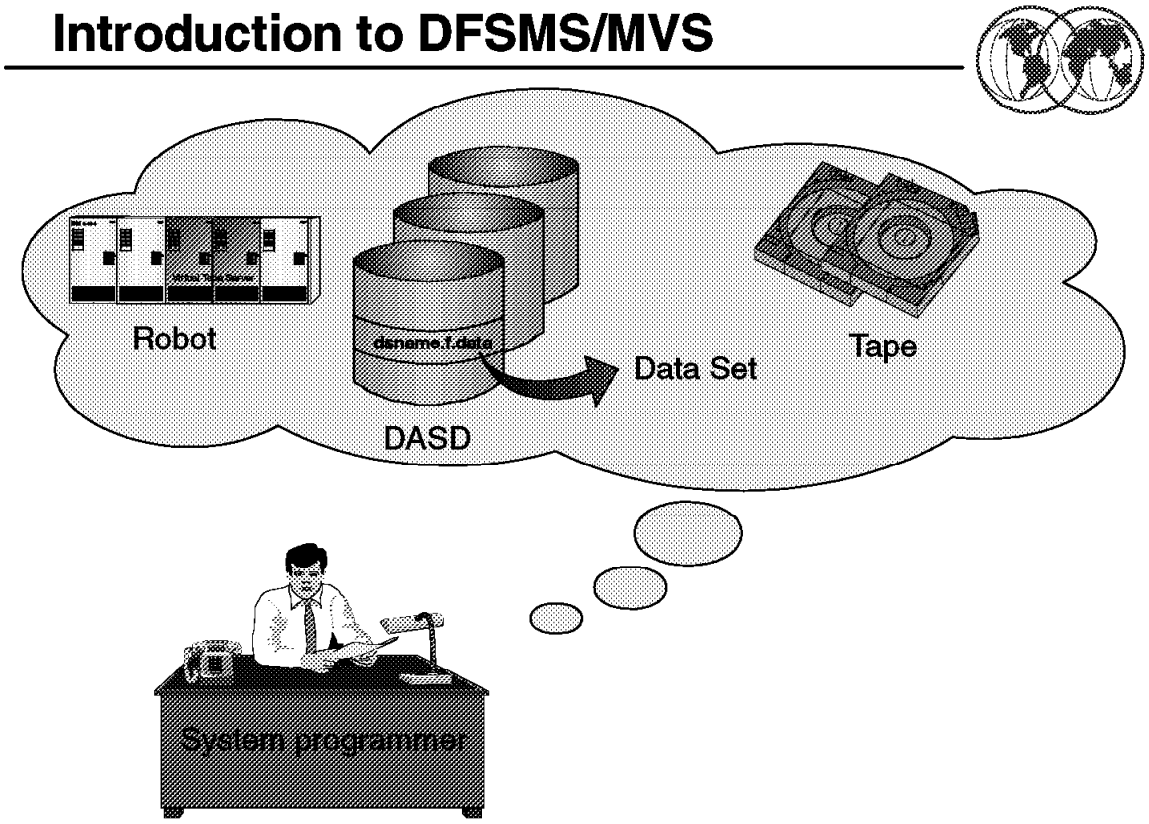


Figure 1. Introduction to data management

1.1 Introduction to data management

Data management is the part of the operating system that organizes, identifies, stores, catalogs, and retrieves all the data information (including programs) that your installation uses. Data management does these main tasks:

- Sets aside (allocates) space on DASD volumes
- Automatically retrieves cataloged data sets by name
- Controls access to data

One of the elements of data management is the *access methods* component, to be described in the next visuals.

This chapter describes MVS data management when processing different types of *data sets*. Also, some comments about how you should name data sets are included.

DFSMS/MVS is a set of products associated with OS/390 that is responsible for data management. DFSMS/MVS has four MVS data management functional components as a single, integrated software package:

DFSMSdfp Provides storage, data, program, and device management. It is made of several components such as access methods, OPEN/CLOSE/EOV routines, catalog management, DADSM (DASD space control), utilities, IDCAMS, SMS, NFS, ISMF, and other functions.

DFSMSdss Provides data movement, copy, backup, and space management functions.

DFSMShsm Provides backup, recovery, migration, and space management functions. It invokes DFSMSdss for certain of its functions.

DFSMSrmm Provides management functions for removable media such as tape cartridges, 3420 reels, and optical media

Before we discuss DFSMS/MVS components, let′s briefly talk about data sets, data organization, volume organization, and data management.

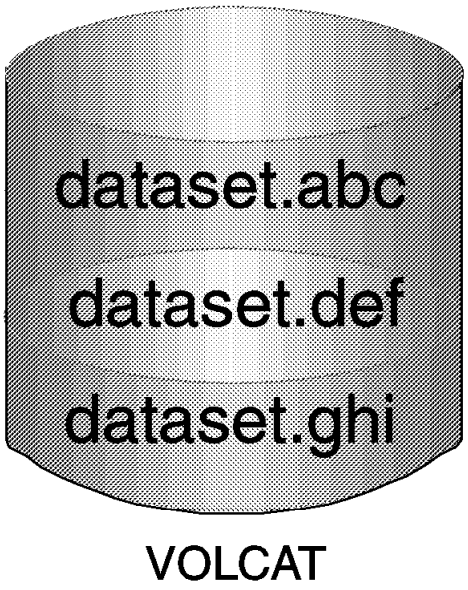


Figure 2. Data sets

1.2 Data sets

An MVS data set is a collection of logically related data records stored on one volume or a set of volumes. A data set can be, for example, a source program, a library of macros, or a file of data records used by a processing program. You can print a data set or display it on a terminal. The logical record is the basic unit of information used by a processing program.

As an exception, the OS/390 UNIX services component supports Hierarchical File Systems (HFS) data sets, where the collection is of bytes and there is not the concept of logically related data records.

Data can be stored on a direct access storage device (DASD), magnetic tape volume, or optical media. The term “DASD” applies to disks or simulated equivalents of disks. All types of data sets can be stored on DASD, but only sequential data sets can be stored on magnetic tape. We discuss the types of data sets later.

The next visuals discuss the logical attributes of a data set which are specified at data set allocation time in:

- DCB/ACB control blocks in the application program
- DD card (explicitly or through Data Class(DC) option)
- In ACS Data Class (DC) routine (overridden by DD card)

After the allocation, such attributes are kept in catalogs and VTOCs.

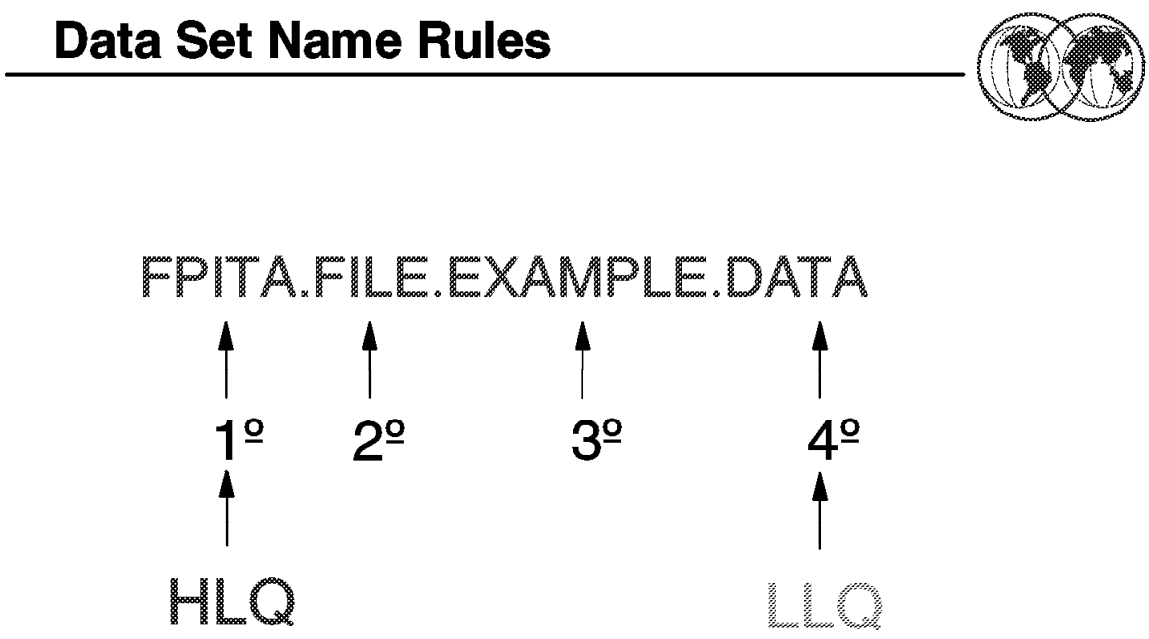


Figure 3. Data set name rules

1.2.1 Data set name rules

Whenever you allocate a new data set, you (or MVS) must give the data set an unique name. Usually, the data set name is given as the DSNAME keyword in JCL.

A data set name can be one name segment, or a series of joined name segments. Each name segment represents a level of qualification. For example, the data set name VERA.LUZ.DATA is composed of three name segments. The first name on the left is called the highest-level qualifier, the last is the lowest-level qualifier.

Each name segment (qualifier) is one to eight characters, the first of which must be alphabetic (A to Z) or national (# @ $). The remaining seven characters are either alphabetic, numeric (0-9), national, or a hyphen (-).

The period (.) separates name segments from each other. Including all name segments and periods, the length of the data set name must not exceed 44 characters. Thus, a maximum of 22 name segments can make up a data set name.

You should only use the low-level qualifier *GxxxxVyy*, structure name, where *xxxx* and *yy* are numbers, in the names of generation data sets (to be seen later). You can define a data set with GxxxxVyy as the low-level qualifier of non-generation data sets only if a generation data group with the same base name does not exist. However, we recommend that you restrict GxxxxVyy qualifiers to generation data sets, to avoid confusing generation data sets with other types of non-VSAM data sets.

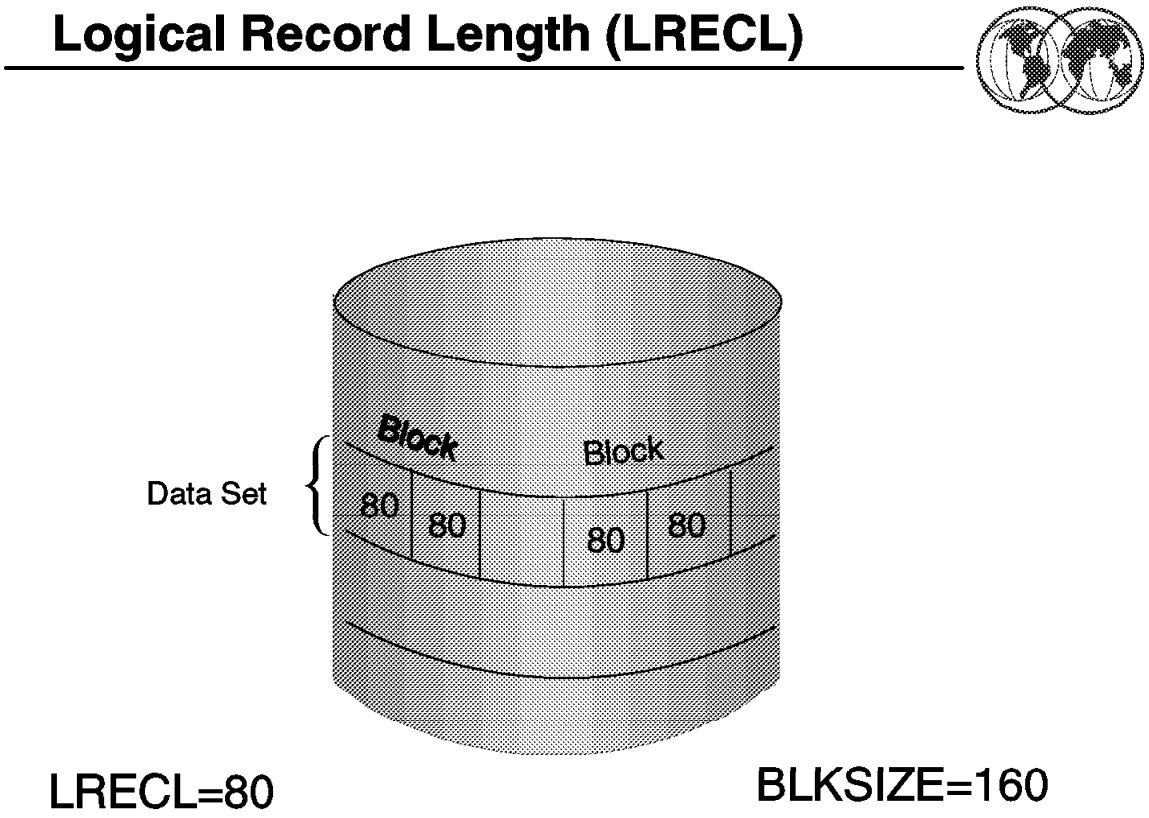


Figure 4. Logical record length

1.2.2 Logical records

A Logical Record is a unit of information about a unit of processing (a customer, an account, a payroll employee). It is the smallest amount of data to be processed and it is made of fields which contain information recognized by the processing application. Logical records when located in DASD, tape, or optical devices are grouped in physical records named blocks. Each block of data on a DASD volume has a distinct location and a unique address, making it possible to find any block without extensive searching. Logical records can be stored and retrieved either directly or sequentially. DASD volumes are used for storing data and executable programs, including the operating system itself, and for temporary working storage. One DASD volume can be used for many different data sets, and space on it can be reallocated and reused.

The maximum length of a logical record (LRECL) is limited by the physical size of the used media.

1.2.3 Record formats

Use the RECFM parameter to specify the format and characteristics of the logical records in a new data set. We may say that they are blocked (several logical records in one block), or no imbedded short blocks, or the existence of an ANSI control character, and so on.

For further information on the REFCM parameter, refer to *DFSMS/MVS Using Data Sets*, SC26-4922, and *OS/390: MVS JCL Reference*, GC28-1757.

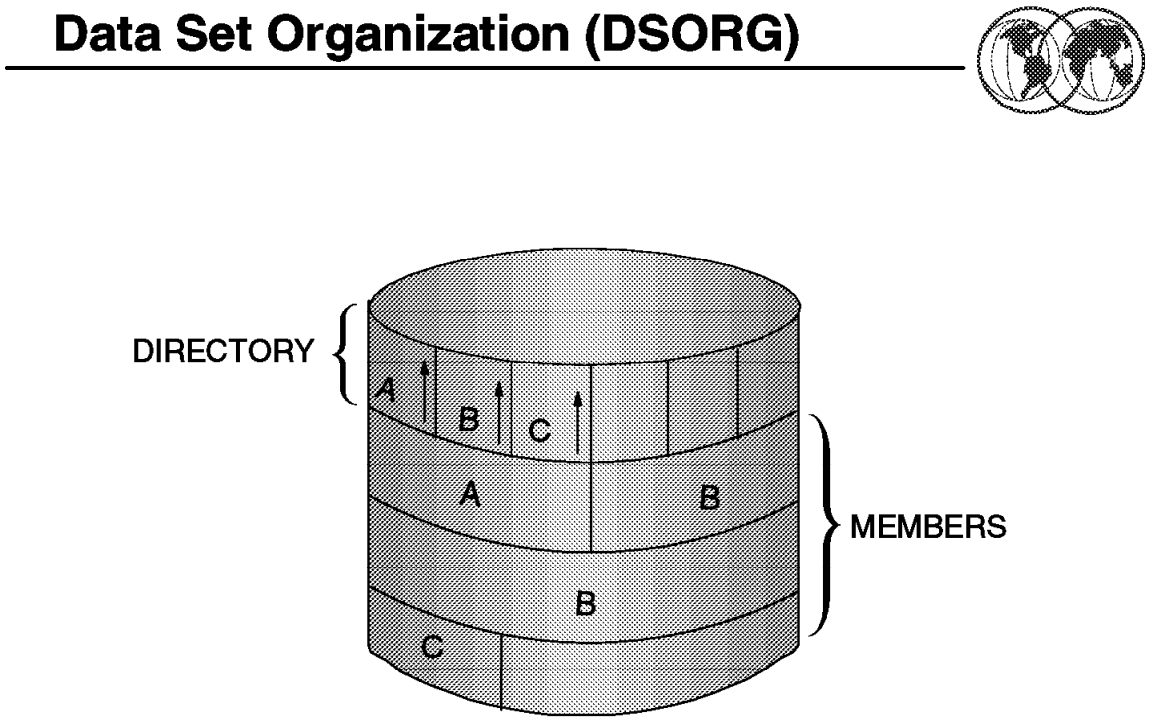


Figure 5. PO data set

1.2.4 Data set organization (DSORG)

There are several different types of data set organization used in OS/390. Each organization provides specific benefits to its user:

1.2.4.1 Physical sequential (PS)

With this data set organization, the records can only be read or written in “physical sequential” order. If we compare this with a PC file, this is a file in the main directory (C:\).

Sequential data sets can exist in DASD, tape, and optical devices.

1.2.4.2 Partitioned Organized (PO)

Partitioned data sets are similar in organization to a library and are often referred to this way. A library contains normally a great number of “books,” and sorted directory entries are used to locate them.

In PDS (partitioned organized data set) the “books” are called *members* and to locate them, they are pointed to by entries in a *directory*, as shown in this visual.

The members are individual sequential data sets and can be read or written sequentially, once they have been located via directory. It is almost the same idea as the directory and file organization in a PC.

Partitioned data sets can only exist on DASD.

Each member has a unique name, one to eight characters long, stored in a directory that is part of the data set. The records of a given member are written or retrieved sequentially. See *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913, for the macros used with partitioned data sets.

The main advantage of using a partitioned data set is that, without searching the entire data set, you can retrieve any individual member after the data set is opened. For example, in a program library (always a partitioned data set) each member is a separate program or subroutine. The individual members can be added or deleted as required. When a member is deleted, the member name is removed from the directory, but the space used by the member cannot be reused until the data set is reorganized; that is, compressed using the IEBCOPY utility (generally requested through an ISPF panel). We discuss IEBCOPY and other DFSMS/MVS utilities later.

The directory, a series of 256-byte records at the beginning of the data set, contains an entry for each member. Each directory entry contains the member name and the starting location of the member within the data set, as shown. Also, you can specify as many as 62 bytes of information in the entry. The directory entries are arranged by name in alphanumeric collating sequence. Each directory block contains a two-byte count field that specifies the number of active bytes in a block (including the count field). Each block is preceded by a hardware defined key field containing the name of the last member entry in the block, that is, the member name with the highest binary value.

Partitioned data set member entries vary in length and are blocked into the member area.

If you do not specify a block size (BLKSIZE), the Open routine determines an optimum block size for you. Therefore, you no longer need to perform calculations based on track length. When you allocate space for your data set, you can specify the average record length in kilobytes or megabytes by using the SPACE and AVGREC parameters, and have the system use the block size it calculated for your data set.

Another type of PO data set is the PDSE, that must be SMS-managed and we will talk about its advantages later.

Refer to 1.18, “Access method functions” on page 85 for more types and more information about data set organization.

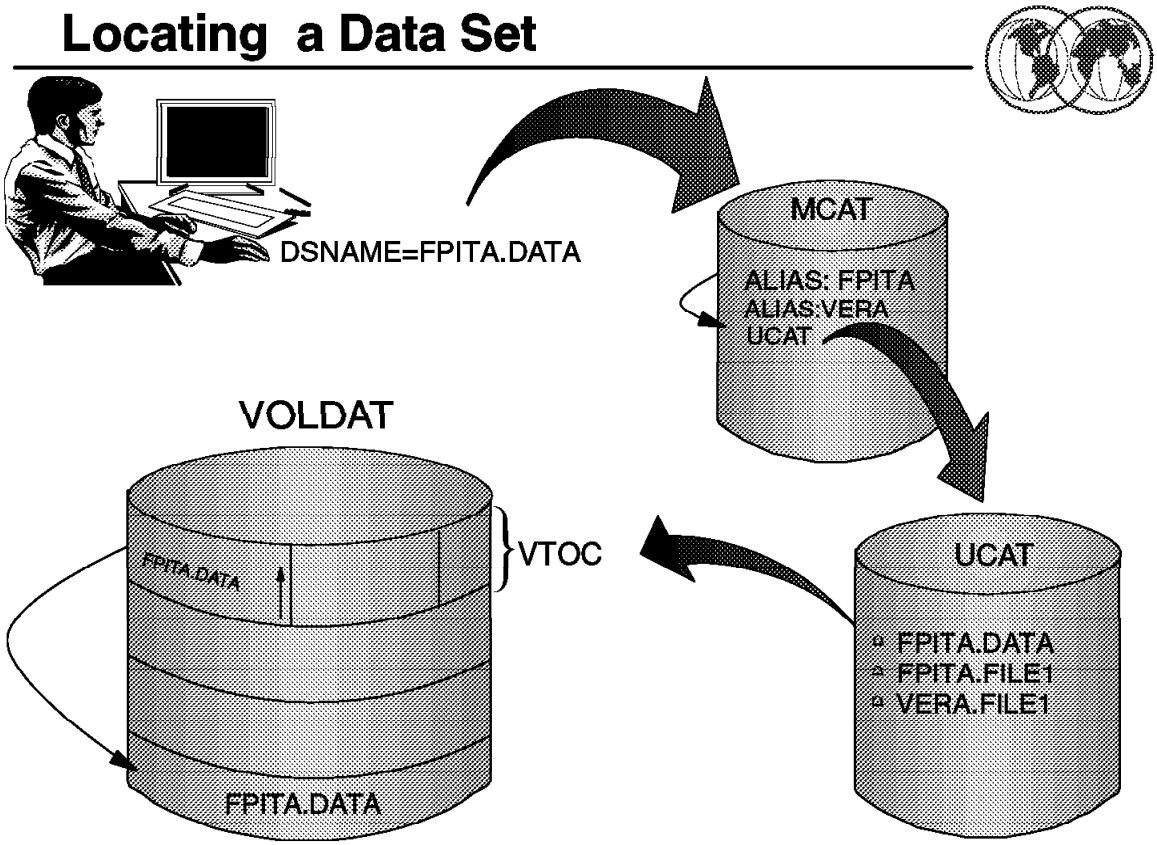


Figure 6. Locating a data set

1.2.5 Locating a data set

Before we explain the procedure used to find a data set, let's introduce some terms used and that will be explained in more detail later.

VTOC Is a sequential data set located in a DASD volume that describes the contents of this volume.

User Catalogs (UCAT) It is a catalog of data sets used to locate in which DASD volume the requested data set is stored; user data sets are are cataloged in this type of catalog.

Master Catalog (MCAT) It has the same structure as a user catalog, but points to system data sets, usually with a high level qualifier (HLQ) name of SYS1. It also contains information about the user catalog location and any alias pointer.

Alias It is an special entry in the Master Catalog pointing to an User Catalog which coincides with the HLQ of a data set. It means that the data set with this HLQ is maybe cataloged in that User Catalog. Then, the alias is used to find in which User Catalog there is that data set location information.

Follows the standard location sequence caused by a request for an already existent data set:

- MCAT is searched, if found, verify if it is:
  - A data set name, then pick up the volume specification and if the indicated device is online, then check VTOC to locate the data set in the specified volume.

  - An alias, that is, the HLQ of the data set name is equal to an alias entry pointing to an UCAT, in this case go to the referred UCAT.

- UCAT is searched (if there is a match in the alias). If the data set name is found, proceed as in an MCAT hit.

Finally, the requiring program can access the data set.

There is another way of using catalogs, where you do not follow the standard location sequence. It is by the use of DD cards named STEPCAT and JOBCAT introducing private catalogs. Use the JOBCAT DD statement to define a private VSAM or user catalog for the duration of a job (step for STEPCAT). The system searches the private catalog for data sets before it searches the master catalog or a user catalog associated with the first qualifier of a data set's name.

It is not recommended that you use private catalogs. One of the reasons is that for SMS-managed data sets SMS only accesses SMS-managed data sets that are cataloged in a system catalog.

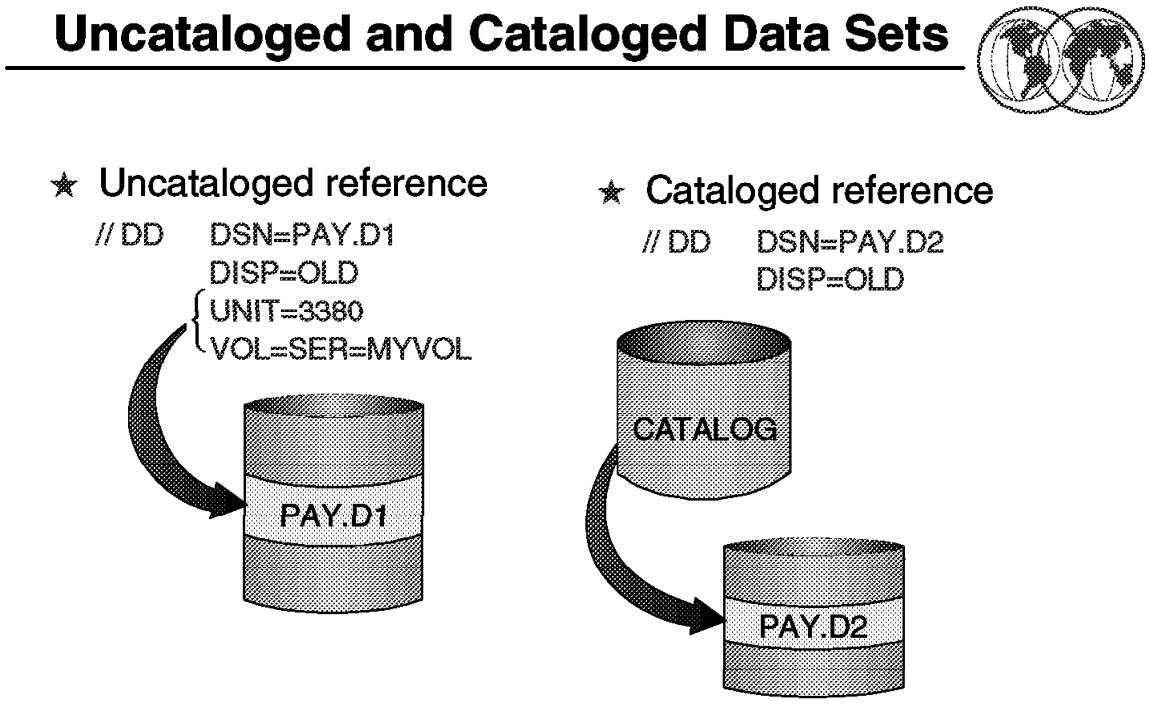


Figure 7. Cataloged and uncataloged data sets

1.2.6 Cataloged and uncataloged data sets

When the data set is cataloged, the system obtains unit and volume information from the catalog. However, if the DD statement for a catalog data set contains VOLUME=SER=serial-number, the system does not look in the catalog; in this case, you must code the UNIT parameter.

When your data set is not cataloged you *must* know in advance its volume location and specify it in your JCL. This can be done through the UNIT and VOL=SER as shown in this visual.

See *OS/390: MVS JCL Reference*, GC28-1757, for information about the UNIT and VOL parameters.

We strongly recommend that you do not have uncataloged data sets in your installation because uncataloged data sets can cause problems with duplicate data and possible incorrect data set processing.

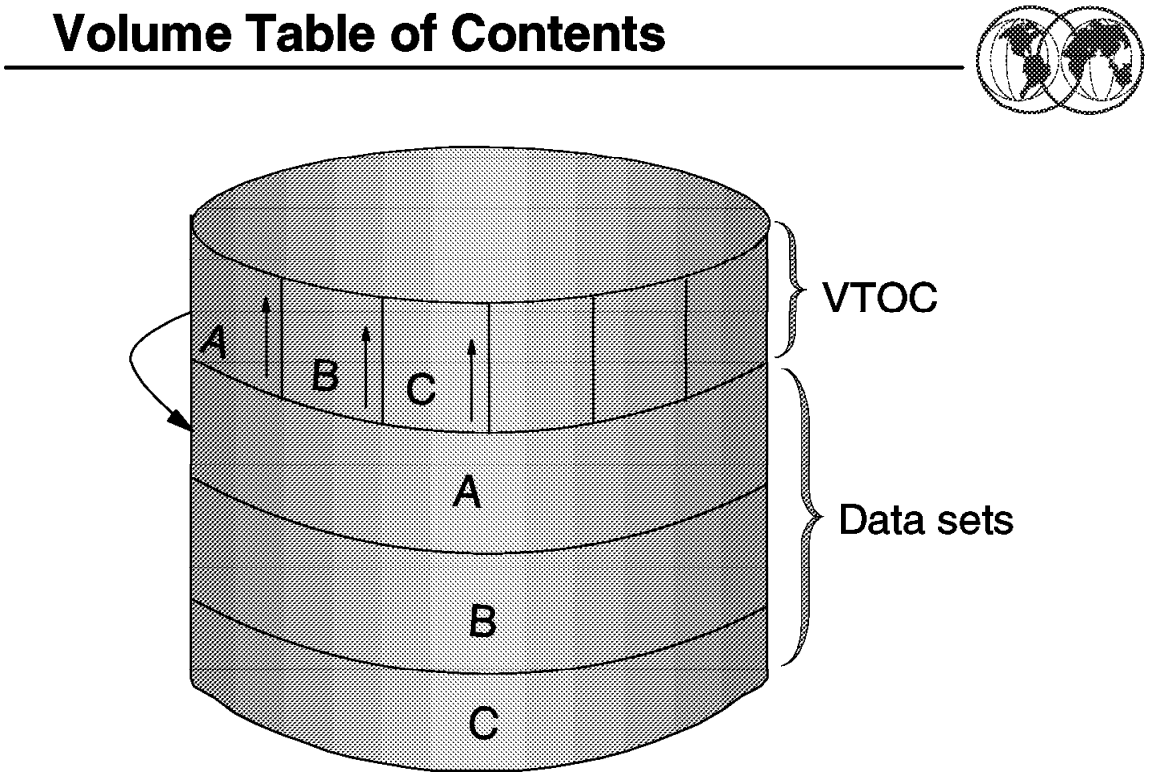


Figure 8. Volume table of contents (VTOC)

1.3 Volume Table of Contents (VTOC)

The VTOC is a data set that describes the contents of the DASD volume on which it resides. It is a contiguous data set; that is, it resides in a single extent on the volume. It is pointed at by the record in the first track of the volume. Data is organized in physical blocks preceded by the highest record key in the block. That is, a count-key-data format.

The VTOC has six types of control blocks, they are called DSCB and describe data set characteristics, free space, and other functions that we will see in the next visuals.

There are a set of macros called Common VTOC Access Facility (CVAF) which allow a program to access VTOC information data.

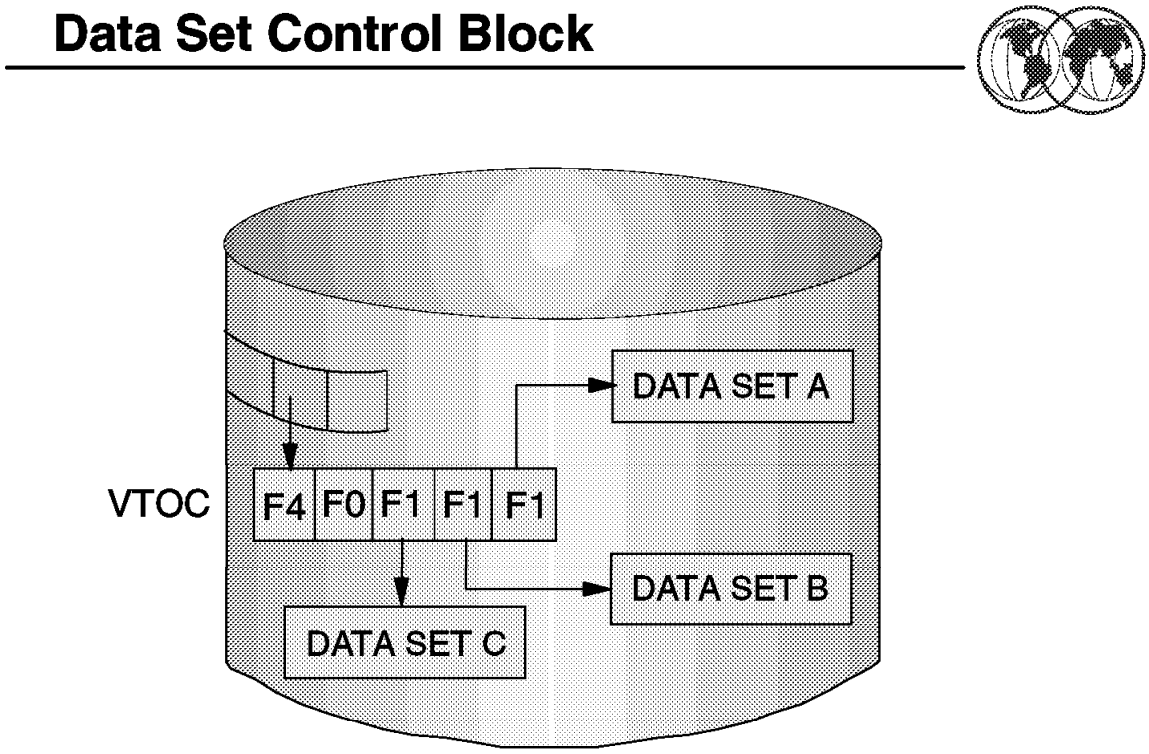


Figure 9. Data set control block (DSCB)

1.3.1 Data set control block (DSCB) types

DSCB is the name of the logical record within the VTOC. DSCBs describe data sets allocated in that volume and also describe the VTOC itself. The system automatically constructs a DSCB when space is requested for a data set on a direct access volume. Each data set on a DASD volume has one or more DSCBs to describe its characteristics. The DSCB appears in the VTOC and, in addition to space allocation and other control information, contains operating system data, device-dependent information, and data set characteristics. There are seven kinds of DSCBs, each with different purpose and a different format number.

The first record in every VTOC is the VTOC DSCB (format-4). The record describes the device, the volume the data set resides on, the volume attributes, and the size and contents of the VTOC data set. The next DSCB in the VTOC data set is a free-space DSCB (format-5) even if the free space is described by format-7 DSCBs. The third and subsequent DSCBs in the VTOC can occur in any order.

Table 1 (Page 1 of 2). DSCBs that can be found in the VTOC

Type	Name	Function	How many
0	Free VTOC DSCB	Describes unused DSCBs in the VTOC (contains 140 bytes of binary zeros).	One for every unused 140-byte record on the VTOC.
1	Identifier	Describes the first three extents of a data set or VSAM data space.	One for every data set or data space on the volume, except the VTOC.

Table 1 (Page 2 of 2). DSCBs that can be found in the VTOC

Type	Name	Function	How many
2	Index	Describes the indexes of an ISAM data set.	One for each ISAM data set (for a multivolume ISAM data set, format-2 DSCB exists only on the first volume). ISAM is an access method generally not used in modern MVS installations.
3	Extension	Describes extents after the third extent of a non-VSAM data set or a VSAM data space.	One for each data set or VSAM data space on the volume that has more than three extents.
4	VTOC	Describes the extent and contents of the VTOC and provides volume and device characteristics. This DSCB contains a flag indicating whether the volume is SMS-managed.	One on each volume.
5	Free Space	On a non-indexed VTOC, describes the space on a volume that has not been allocated to a data set or to a VSAM data space (available space). For an indexed VTOC, a single empty format-5 DSCB resides in the VTOC;	One for every 26 non-contiguous extents of available space on the volume for a non-indexed VTOC; for an indexed VTOC, there is only one.
7	Free space for certain device	Only one field in the format-7 DSCB is an intended interface. This field indicates whether the DSCB is a format-7 DSCB. You can reference that field as DS1FMTID or DS5FMTID. A character 7 indicates that the DSCB is format-7 DSCB, and your program should not modify it. If you are diagnosing a problem, see *DFSMS/MVS DFSMSdfp Diagnosis Guide*, SY27-9605, for the layout of the Format-7 DSCB.	

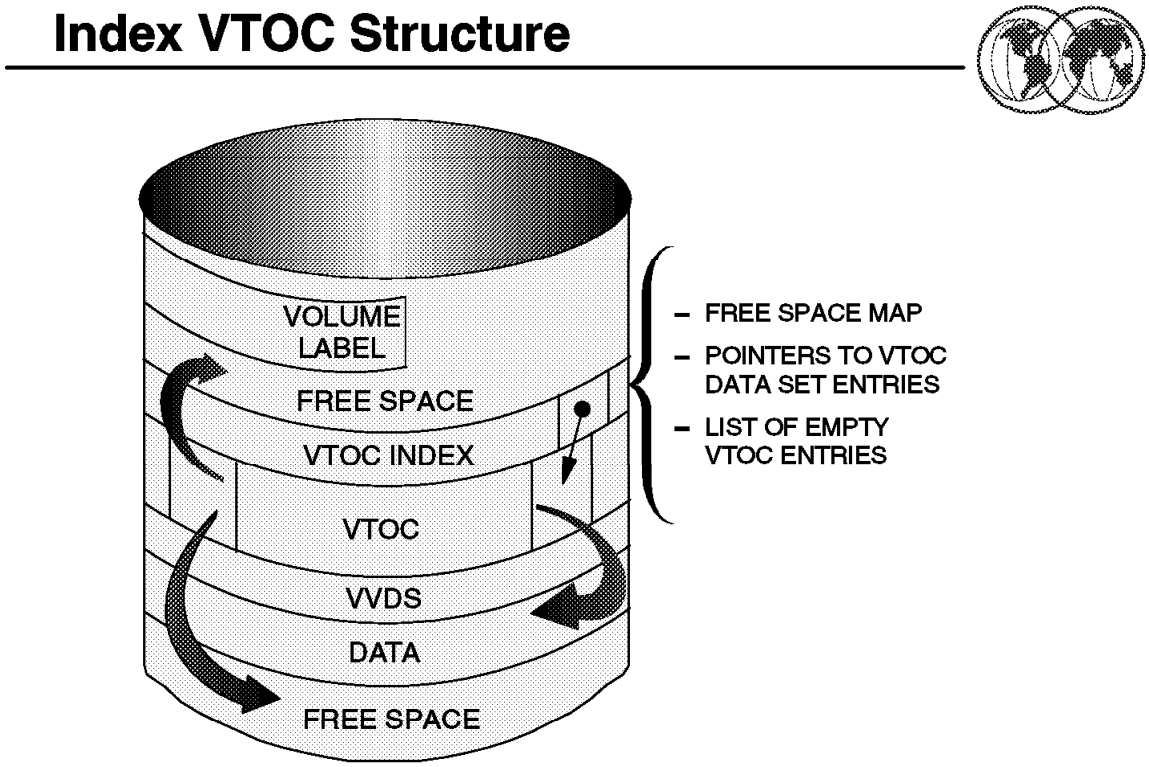


Figure 10. Index VTOC structure

1.3.2 Index VTOC structure

The Index VTOC may provide poor performance, mainly when many data sets are located in that volume. The major reason is the lack of an index to speed up the search. Optionally an index VTOC can be associated with the VTOC. The index VTOC enhances the performance of VTOC access. The VTOC index is a physical-sequential data set on the same volume as the related VTOC. It consists of an index of data set names in format-1 DSCBs contained in the VTOC and volume free-space information.

Note: An SMS-managed volume requires an indexed VTOC; otherwise, the VTOC index is highly recommended

In MVS/DFP 3.3 the index VTOC was improved to make more efficient use of space in an index.

1.3.3 Creating the VTOC and index VTOC

To initialize a volume (preparing for I/O activity), use the Device Support Facilities (ICKDSF) utility to initially build the VTOC. You can create a index VTOC at that time, by using the ICKDSF INIT command and specifying the INDEX keyword.

You may use ICKDSF to convert a non-indexed VTOC to an indexed VTOC by using the BUILDIX command and specifying the IXVTOC keyword. The reverse operation can be performed by using the BUILDIX command and specifying the OSVTOC keyword. See the *ICKDSF R16 User's Guide*, GC35-0033, for details and refer to *DFSMS/MVS DFSMSdfp Advanced Services*, SC26-4921.

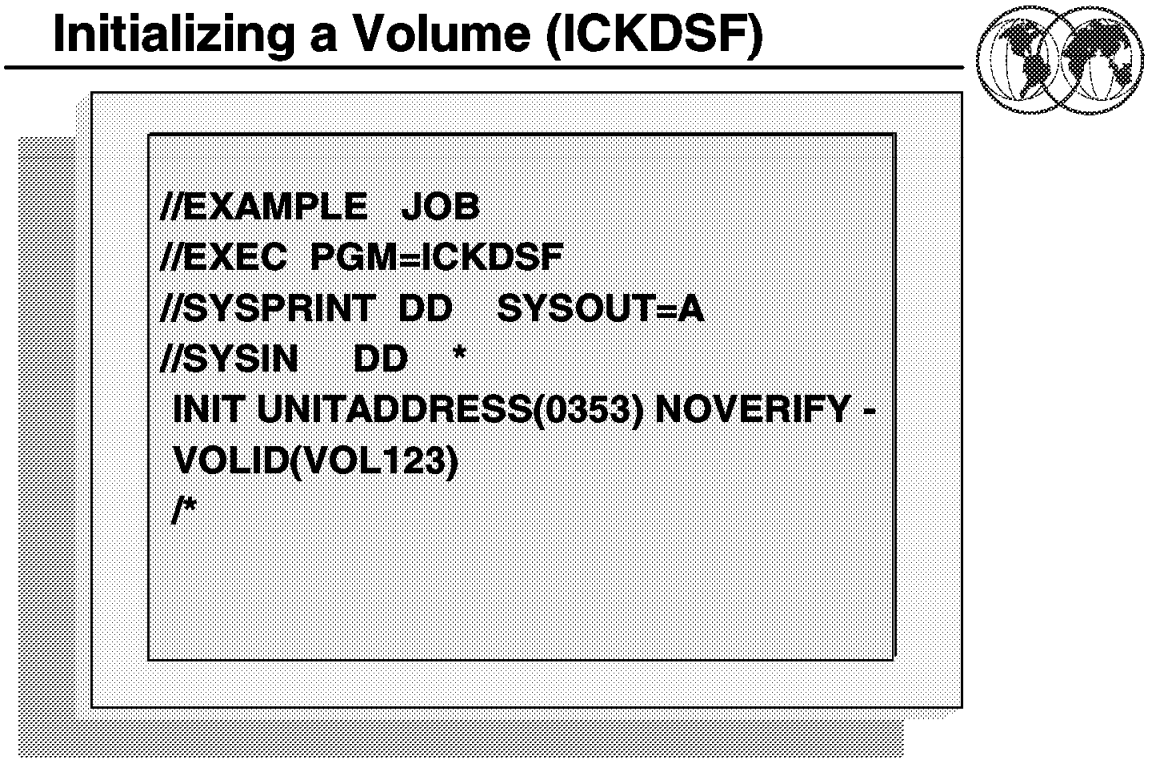


Figure 11. Initializing a volume

1.4 ICKDSF

ICKDSF is a program you can use to perform functions needed for the initialization, installation, use, and maintenance of IBM DASD. You can also use it to perform service functions, error detection, and media maintenance.

On the modern DASD devices there is not a reason to run error detection and media maintenance. These functions are supported internally by the controller. On the other hand the concept of an MVS volume is not mapped into a unique physical DASD Redundant Access of Independent Disks (RAID) volume. Due to RAID the MVS volume may be spread in several little size disks as in the case of RVA virtualization. Then, in the following examples do not take in consideration Analyze and Inspect if you have a DASD more modern than a real 3390.

1.4.1 Initializing a DASD volume

After you have completed the installation of a device, you must initialize and format the volume so that it can be used by MVS. If the volume is SMS-managed, the the STORAGEGROUP option must be declared.

1.4.2 VTOC and index VTOC

The INIT and BUILDIX commands will build the VTOC index. The INIT command creates space for the index during volume initialization in both operating system and stand-alone versions. The BUILDIX command, which requires that the host operating system contain indexed VTOC programming support, builds VTOC indexes on volumes current in use on the system. Both commands prepare the VTOC on the target volume to indexed VTOC (IXVTOC) format.

1.4.2.1 INIT examples

Following are some examples of initializing volumes.

Initializing a volume for the first time in offline mode

In this example, a volume is initialized at the minimal level because neither the CHECK nor VALIDATE parameter is specified. Because the volume is being initialized for the first time, it must be mounted offline, and the volume serial number must be specified. Because the VTOC parameter is not specified, the default volume table of contents size is the number of tracks in a cylinder minus one. For a 3390, the default is cylinder 0, track 1 for 14 tracks.

```
//EXAMPLE   JOB
//          EXEC  PGM=ICKDSF
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
 INIT UNITADDRESS(0353) NOVERIFY VOLID(VOL123) -
      OWNERID(PAYROLL)
/*
```

Initializing a volume to be managed in a DFSMS environment

In the following example, a volume that is to be system-managed is initialized. The volume is initialized in offline mode at the minimal level. The VTOC is placed at cylinder 2, track 1 and occupies ten tracks. The VTOC is followed by the VTOC index. The STORAGEGROUP parameter indicates the volume is to be managed in a DFSMS environment.

```
INIT UNIT(0353) NOVERIFY STORAGEGROUP -
       OWNERID(PAYROLL) VTOC(2,1,10) INDEX(2,11,5)
```

This example performs an online minimal initialization, and as a result of the command, an index to the VTOC is created:

```
//         JOB
//         EXEC  PGM=ICKDSF
//XYZ987   DD    UNIT=3390,DISP=OLD,VOL=SER=PAY456
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
 INIT DDNAME(XYZ987) NOVERIFY INDEX(X'A',X'B',X'2')
/*
```

1.4.3 ICKDSF stand-alone version

You can run the stand-alone version of ICKDSF under any IBM S/390 machine.

To run the stand-alone version of ICKDSF under an IBM S/390, you IPL ICKDSF with a stand-alone IPL tape that you create under MVS.

1.4.3.1 Creating an ICKDSF stand-alone IPL tape using MVS

For MVS, the stand-alone code is in SYS1.SAMPLIB as ICKSADSF. You can load the ICKDSF program from a file on tape. The following example can be used to copy the stand-alone program to an unlabeled tape:

```
//JOBNAME  JOB  JOB CARD PARAMETERS
//STEPNAME EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN    DD DUMMY,DCB=BLKSIZE=80
//SYSUT1   DD DSNAME=SYS1.SAMPLIB(ICKSADSF),UNIT=SYSDA,
//            DISP=SHR,VOLUME=SER=XXXXXX
//SYSUT2   DD DSNAME=ICKDSF,UNIT=3480,LABEL=(,NL),
//            DISP=(,KEEP),VOLUME=SER=YYYYYY,
//            DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
```

For details on how to IPL the stand-alone version and examples of the commands, refer to *Device Support Facilities User's Guide and Reference Release 16 Refresh*, GC35-0033.

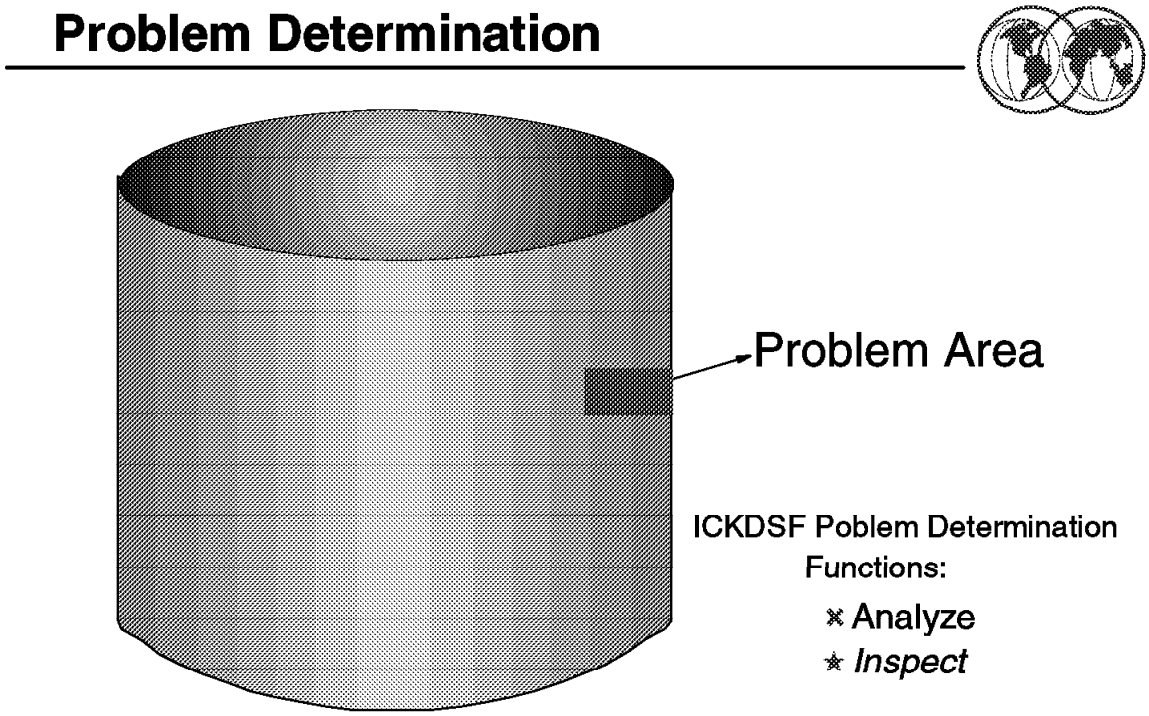


Figure 12. Problem determination

1.4.4 Problem determination

You can use ICKDSF to help determine if the origin of a problem is hardware or recording media.

1.4.4.1 Data check

A data check is an error detected in the bit pattern read from the disk. When it is a media problem it is most likely caused by an imperfection on the disk surface.

1.4.4.2 Analyze

The ANALYZE command helps to detect and differentiate recording surface and drive-related problems on a volume. It can also scan data to help detect possible media problems.

You can use the ANALYZE command to examine a device and the data on a volume to help determine the existence and the nature of errors.

You use two parameters with the ANALYZE command:

- DRIVETEST tests the hardware device
- SCAN reads data on a volume

You can use the DRIVETEST parameter to ensure that device hardware can perform basic operations, such as seeks, reads, and writes. DRIVETEST can impact your system performance, but does not alter data.

You can use ANALYZE SCAN to read data that currently exists on a volume. If ANALYZE SCAN reads the data successfully the first time, no further rereading of the track takes place.

1.4.4.3 INSPECT command

The INSPECT command inspects a subset of a volume and can:

- Check the surface of a track to determine if there is a defect
- Assign a skip to avoid a defect
- Assign an alternate track
- Reclaim a track that has been flagged defective
- Print a map of defective tracks on a volume

Before using the INSPECT command, you should first make sure there are no hardware problems. It is recommended that you issue ANALYZE DRIVETEST NOSCAN before any INSPECT operation.

For more information about ICKDSF, refer to *Device Support Facilities User's Guide and Reference Release 16 Refresh*, GC35-0033.

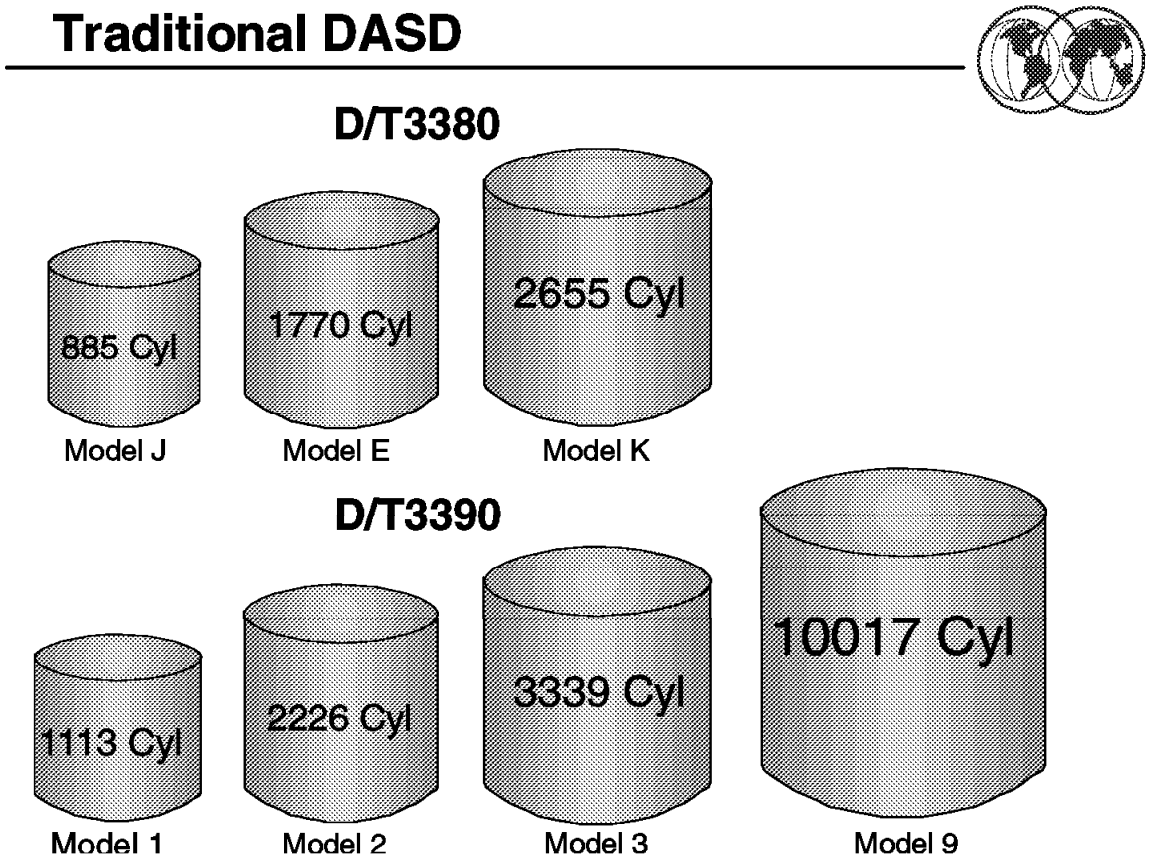


Figure 13. Traditional DASD

1.5 Traditional DASD

Traditional DASD means 3380 and 3390 type of devices. The more modern IBM DASD products such as RAMACs, RVA, and Enterprise Storage Server (Shark) including OEM DASD emulates IBM 3380 and 3390 volumes in the geometry, capacity of track, and number of tracks per cylinder.

Table 2 shows further information about DASD capacity.

Table 2. DASD capacity

Physical characteristics	**3380-J**	**3380-E**	**3380-K**	**3390-1**	**3390-2**	**3390-3**	**3390-9**
Data Cyl/Device	885	1770	2655	1113	2226	3339	10017
Track/cyl	15	15	15	15	15	15	15
Bytes/trk	47476	47476	47476	56664	56664	56664	56664
Bytes/Cylinder	712140	712140	712140	849960	849960	849960	849960
MB/Device	630	1260	1890	946	1892	2838	8514

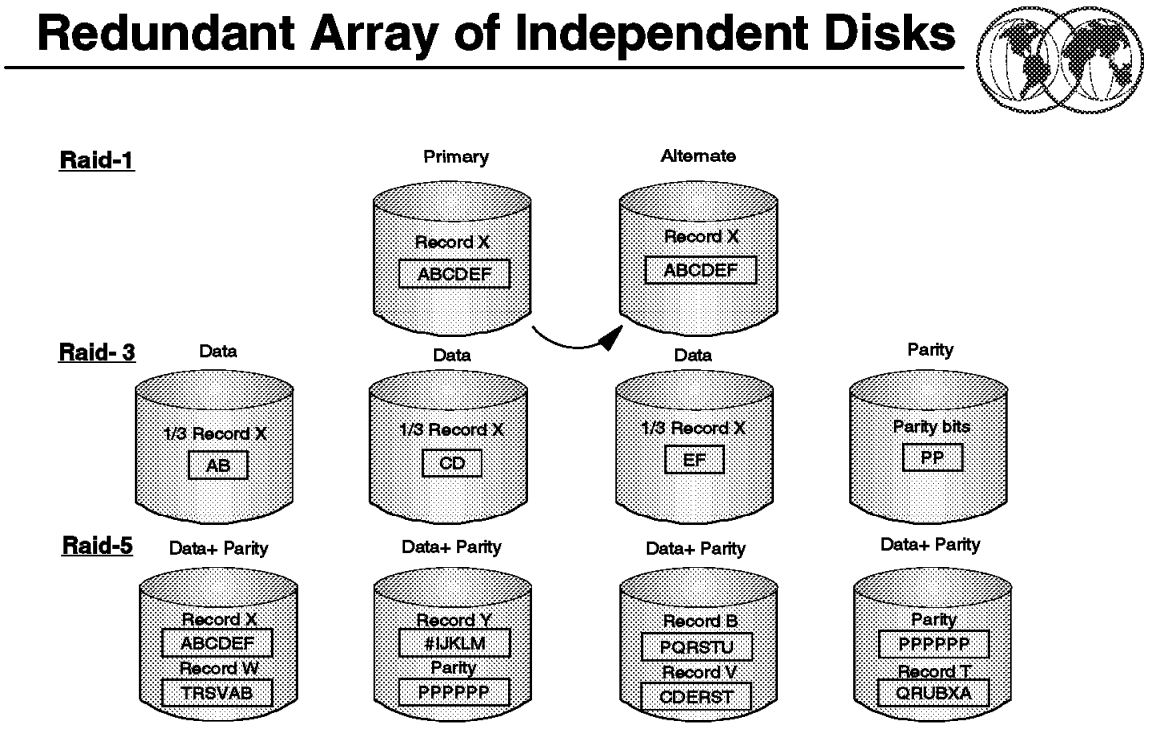


Figure 14. Redundant array of independent disks (RAID)

1.6 Redundant Array of Independent Disks (RAID)

Redundant array of independent disks (RAID) is a direct access storage architecture where data is recorded across multiple physical disks with parity separately recorded so that no loss of access to data results from the loss of any one disk in the array.

The RAID concept involves many little small computer system interface (SCSI) disks replacing a big one. The major RAID advantages are:

- Performance (due to parallelism)
- Cost (SCSI commodities)
- S/390 compatibility
- Environment (space and energy)

However, RAID increased the chances of malfunction due to media and disk failures and the fact that the logical device is now residing on many physical disks. The solution was redundancy, which wastes space and cause performance problems as “write penalty” and “free space reclamation.”

To address this performance issue, large caches are implemented.

Various implementations certified by RAID Architecture Board are:

RAID-1 Has just disk mirroring like dual copy.

RAID-3 Has an array with one parity device and just one I/O request at time with intra-record striping. The access arms move together. It has a high data rate and a low I/O rate.

RAID-5 Has an array with one distributed parity and has four HDAs in an RAMAC-3 array. It does I/O requests in parallel with extra-record striping. The access arms move independently. It has strong caching to avoid “write penalties”; that is four I/Os per write. RAID-5 has a high I/O rate and a medium data rate. RAID-5 does the following:

- Reads data from an undamaged HDA is just one single I/O operation.
- Reads data from a damaged HDA which implies (n-1) I/Os, where n is the number of HDAs in the array operation.
- For every write to an undamaged HDA, RAID-5 does four I/O operations in order to store a correct parity block (write penalty). This penalty can be relieved with strong caching and a slice triggered algorithm (coalescing updates into a single parallel I/O).
- For every write to a damaged HDA, RAID-5 does n-1 reads and one parity write.

RAID-6 RAID-6 has an array with two distributed parity and I/O requests in parallel with extra-record striping. Its access arms move independently (Reed/Salomon P-Q parity). The write penalty is greater than RAID-5 with six I/Os per write.

RAID-6+ RAID-6+ is without write penalty (due to log-structured file, LFS), and has background free-space reclamation. The access arms all move together for writes.

RAID-10 RAID-10 has a new RAID architecture designed to give performance for striping and has redundancy for mirroring.

Note: Data striping is called RAID-0, but it is not a real RAID because of no redundancy.

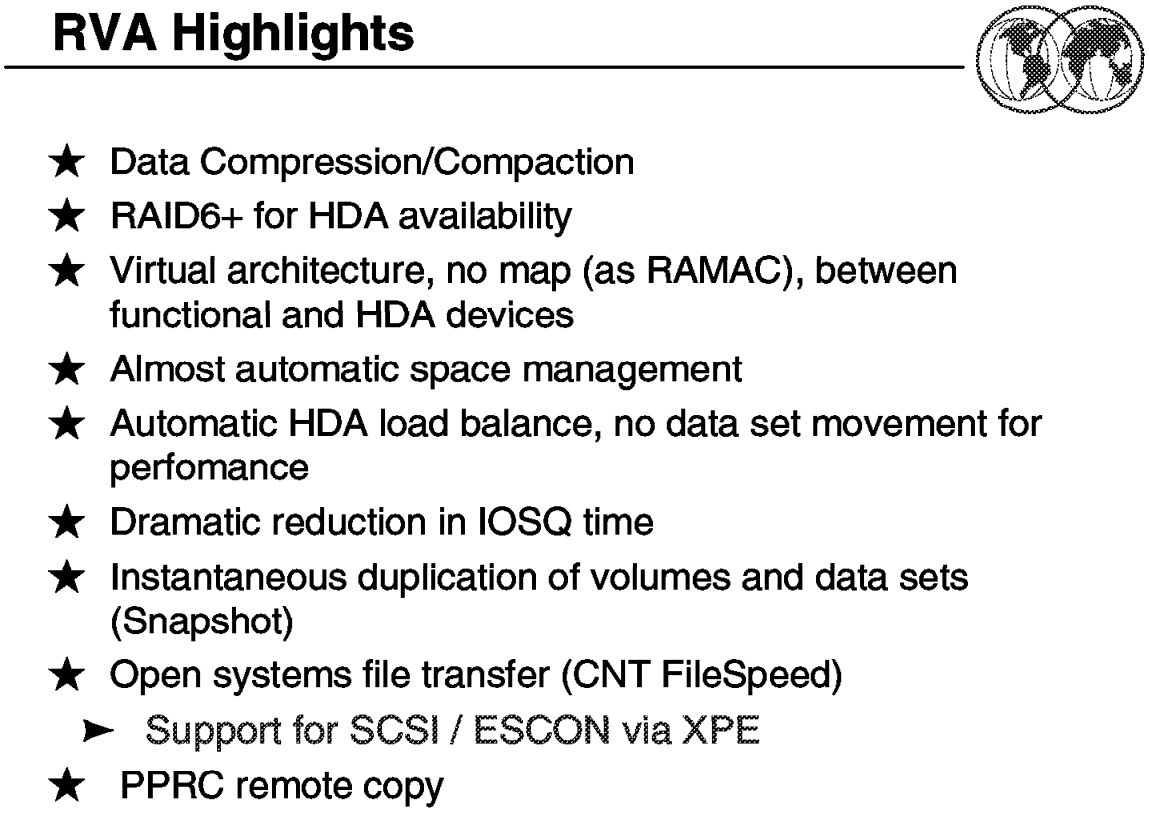


Figure 15. RVA highlights

1.7 RVA highlights

Following is a list of the main properties and functions of the RAMAC Virtual Array (RVA) 9393:

- Controller does data compression and compaction, when data enters the facility. It saves space in cache and DASD media, saving also CPU cycles (if the compression is done in CPU). Usually because the compression factor is greater than 5.0 you get more space than the advertised amount.
- From 160-Gb to 726-Gb of effective capacity per subsystem. Each subsystem has one frame with two to eight trays. Each tray contains seven (or eight) HDAs with two spares per subsystem.
- HDA capacity of 4.5 Gb. IBM Ultrastar 2XP 3.5″ disks, 4.5-Gb SCSI/FBA. Mean Time Between Failures (MTBF) of 1-M hours.
- Models: 002 (ESCON/Parallel), T82 (ESCON).
- RAID6+ allows better availability. You only loose your data if you have more than *two* failures in the disks (HDA) of one string.
- Virtual architecture, a copy of the S/390 virtual storage. Virtual volumes, also called functional devices are mapped in the HDAs.
- Almost automatic space management (due to large number of “virtual volumes”):
  - Reduction or elimination of Defrags.
  - Reduced need to segregate disk pools by file size.

  - Reduced reruns due to out-of-space abends (X37).
  - Reduced VSAM/database reorganizations.
  - Elimination of HSM disk-to-disk migration (ML1).
- Automatic HDA load balance, with no data set movement for performance.
- Reduction in IOSQ time.
- Almost instantaneous duplication of volumes and data sets (Snapshot) without an additional space.
- Open systems file transfer (with CNT FileSpeed).
- Support for SCSI and ESCON via XPE (without the 20 meters collocation problem).
- Simulates all 3390s (9′s is a SOD) and 3380s up to 256.
- Two clusters, eight storage paths with compression engines, cache and device interfaces. Capable of eight concurrent data transfers plus four paths for CCW processing.
- PPRC remote copy implementation.

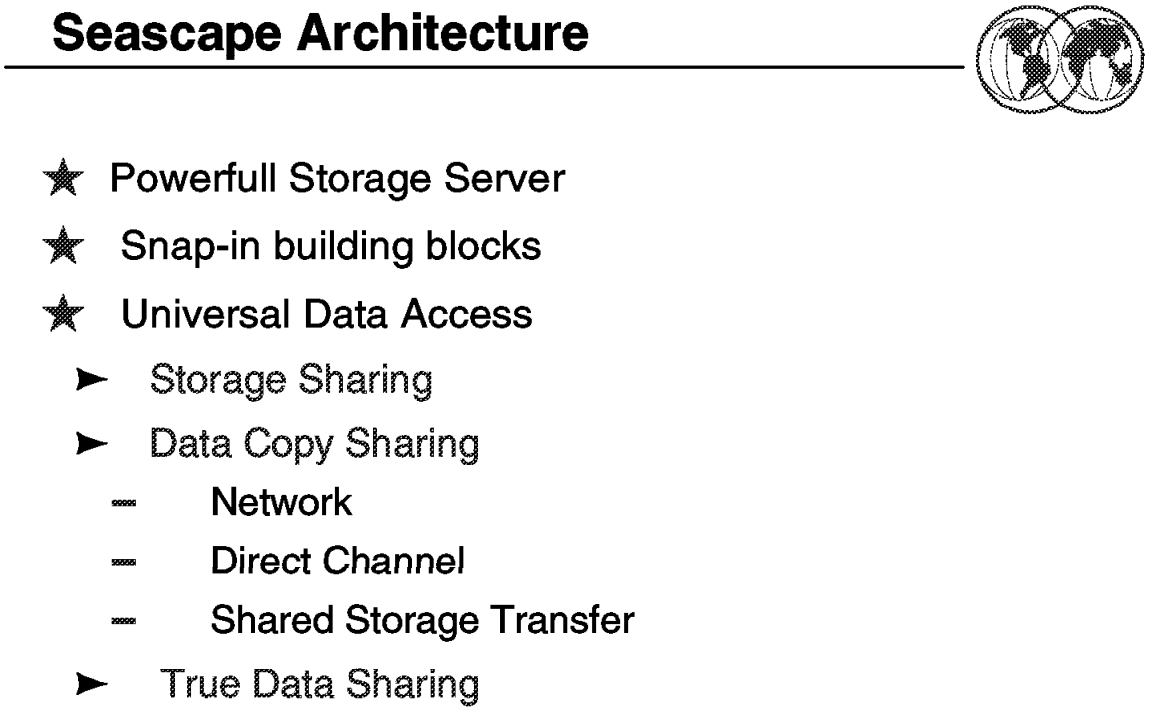


Figure 16. Seascape architecture

1.8 Seascape architecture

Enterprise Storage Server (ESS), shown in Figure 17 on page 29, is the latest IBM storage product using Seascape architecture.

The ESS is also the first of the Seascape architecture storage products to provide attachment to IBM S/390 and open-system platforms. The Seascape architecture products come with integrated storage controllers. These integrated storage controllers allow the attachment of physical storage devices that emulate 3390 Models 2, 3, and 9 or provide 3380 track-compatibility mode.

Seascape is a storage enterprise architecture that is ideally suited to provide storage server solutions for the networked world. Seascape has three basic concepts as follows:

- Powerful storage server
- Snap-in building blocks
- Universal data access

1.8.1 Powerful storage server

It has a storage system which is intelligent and independent, and which can be reached by channels or via the network.

1.8.1.1 Snap-in building blocks

This is a concept where each Seascape product is made of some building blocks such as:

- Scalable n-way RISC server, PCI based; this provides the logic of the storage server.
- Memory cache from RISC processor memory.
- Channel attachments as FC-AL, SCSI, ESCON, and SSA.
- Network attachments such as Ethernet, FDDI, TR, and ATM.

  These attachments may also implement functions. As you can see, a mix of network interfaces (to be used as a remote and independent storage server) and channel interfaces (to be used as storage controller interface).
- Software building blocks such as AIX subset, ADSM, JAVA applications, and Tivoli. High level language (HLL) is better than microcode in flexibility, easier to write, and maintain.
- Storage adapters for mixed storage devices technologies.
- Storage device building blocks such as serial disk (7133), 3590 tape (Magstar), optical (3995).
- Silos and robots (3494).

1.8.1.2 Universal data access

Universal data access allows a wide array of connectivity such as: WIN, OS/2, UNIX, AS/400, and S/390. There are three types of universal access:

- *Storage sharing*

  Physical storage (DASD or tape) is statically divided into fixed partitions available to a given processor. It is not a software function. The subsystem controller knows which processors own what storage partitions. In a sense, just capacity is shared, not data. One server cannot access the data of the other server. It is required that the manual reassigning of storage capacity between partitions be simple and nondisruptive.

  Advantages:

  - Purchase higher quantities with greater discounts
  - Just one type of storage to manage
  - Static shifting of capacity as needed
- The drawbacks are:
  - Higher price for SCSI data
  - Collocation at 20 meters of the SCSI servers (not true with XPE)
  - No priority concept between S/390 and UNIX/NT I/O requests
- *Data copy sharing*

  Data copy sharing is an interim data replication solution (waiting for a true data sharing) done via data replication through software and hardware.

  There are three ways to implement data copy sharing:

  - Network: Via network data transfer as SNA or TCP/IP.

    Problems: CPU and network overhead, still slow and expensive for massive data transfer.
  - Direct channel: Direct data transfer between the processors involved using channel or bus capabilities, referred as bulk data transfer.

    One example can be the IBM Infospeed where a /390 FTP (PDM - Press Data Mover) at 28-MB/sec extract utility program writes data at 28-MB/sec to a pipe (Infospeed box) that is concurrently being read by UNIX and NT.

  - Shared storage transfer: Writing an intermediate flat file by software into the storage subsystem cache, that is read (and translated) by the receiving processor, so the storage is shared.

- *True data sharing*

  Via multiple platform read/write in a single copy. Address complex issues of mixed data types, file structures, databases, and SCPs.

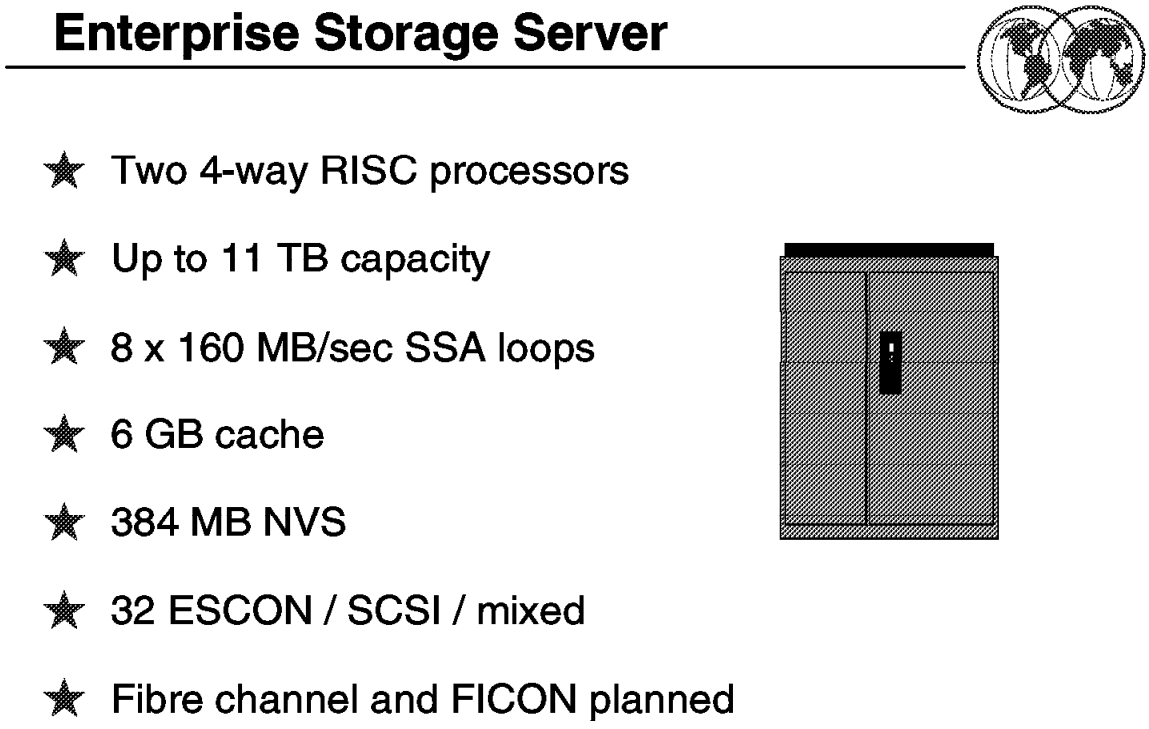


Figure 17. Enterprise Storage Server

1.8.2 Enterprise Storage Server

The IBM Enterprise Storage Server (ESS) is a high performance, high-availability capacity storage subsystem. It contains two four-way RISC processors with 6 GB cache and 384 MB of non-volatile storage to protect from data loss. The ESS maximum capacity is over 11 TB with the second frame attached.

The ESS is an IBM high-end storage subsystem. It is the newest storage subsystem succeeding the 3880 family, the 3990 family, and the 9340 family. Designed for mid-range and high-end environments, the ESS, shown in the visual, provides you with large capacity, high performance, continuous availability, and storage expandability.

The ESS has eight Serial Storage Architecture (SSA) loops, each one with a rate of 160 MB/sec for accessing the disks.

ESS implements RAID-5 for availability and has seven data disks plus one parity disk in the majority of the arrays.

Connectivity to S/390 is through up to 32 ESCON channels, and to UNIX, AS/400, or NT connectivity is through up to 32 SCSI interfaces, or a combination of the two.

The SCSI adapter is a card in the host. It connects to a SCSI bus via an SCSI port. There are two different types of SCSI supported by ESS as follows:

- SCSI Fast Wide with 20 MB/sec

- Ultra SCSI Wide with 40 MB/sec

Comparing the terminology of ESCON and SCSI, we may say that:

- An ESCON channel translates into an SCSI adapter (both cards in the host)
- An ESCON port translates into an SCSI port (both connectors)
- An ESCON link translates into an SCSI bus (both cables)

In the future, ESS will support both S/390 Fiber Channel (FICON) and Fibre Channel Protocol (FCP), including Fiber Channel Arbitrated Loop (FCAL) and FC-switched for systems. Currently you can attach to a FICON channel using the 9032 ESCON Director and to a Fibre Channel network using the IBM SAN Data Gateway. Some of the RVA functions are not implemented, such as:

- Log Structured File (LSF) or virtualization
- Snapshot
- Storage compression

However, there is a statement of direction about its implementation in ESS.

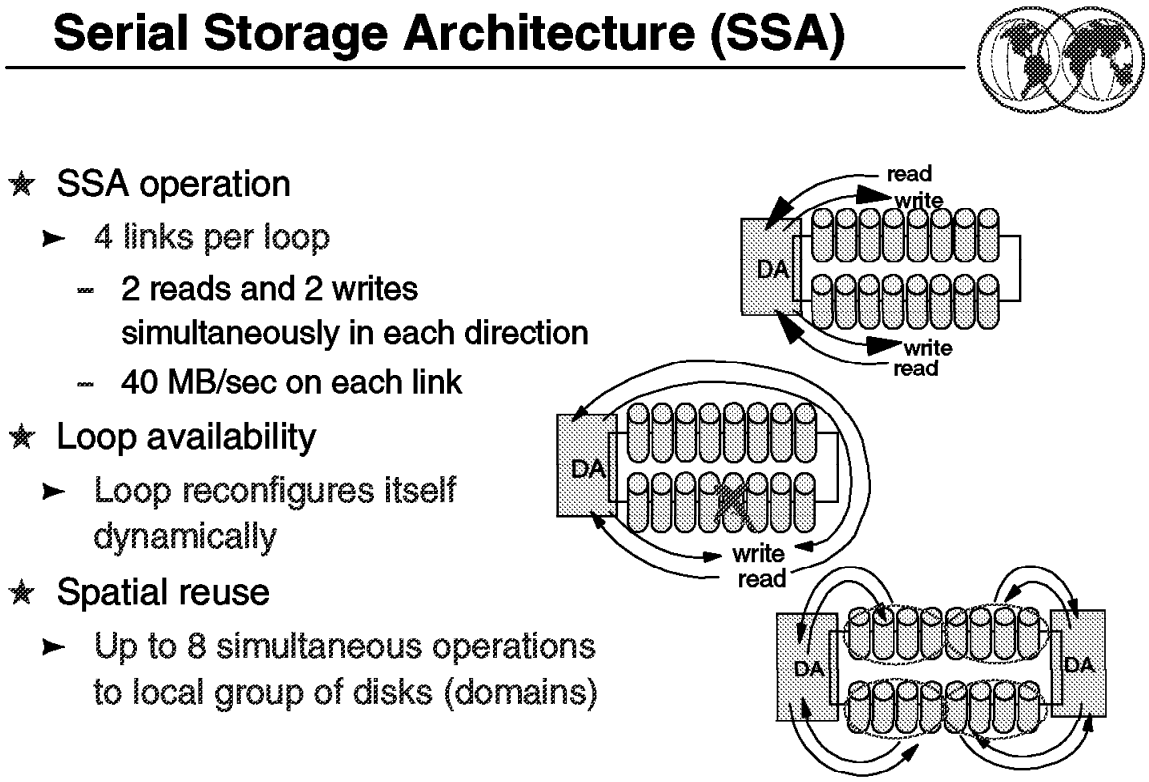


Figure 18. Serial Storage Architecture (SSA)

1.8.3 Serial Storage Architecture (SSA)

SSA is a high performance, serial connection technology for disk drives. SSA is a full-duplex loop-based architecture, with two physical read paths and two physical write paths to every disk attached to the loop. Data is sent from the adapter card to the first disk on the loop and then passed around the loop by the disks until it arrives at the target disk. Unlike bus-based designs, which reserve the whole bus for data transfer, SSA only uses the part of the loop between adjacent disks for data transfer. This means that many simultaneous data transfers can take place on an SSA loop, and it is one of the main reasons that SSA performs so much better than SCSI. This simultaneous transfer capability is known as spatial release.

Each read or write path on the loop operates at 40 MB/s, providing a total loop bandwidth of 160 MB/s.

Loop availability

The loop is a self-configuring, self repairing design which allows genuine hot-plugging. If the loop breaks for any reason, then the adapter card will automatically reconfigure the loop into two single loops. In the ESS, the most likely scenario for a broken loop is if the actual disk drive interface electronics should fail. If this should happen, the adapter card will dynamically reconfigure the loop into two single loops, effectively isolating the failed disk. If the disk were part of a RAID array, the adapter card would automatically regenerate the missing disk using the remaining data and parity disks to the spare disk. Once the failed disk has been replaced, the loop will automatically be reconfigured into full duplex operation, and the replaced disk will become a new spare.

Spatial reuse

Spatial reuse allows domains to be set up on the loop. A domain means that one or more groups of disks belong to one of the two adapter cards, as is the case during normal operation. The benefit of this is that each adapter card can talk to its domains (or disk groups) using only part of the loop. The use of domains allows each adapter card to operate at maximum capability because it is not limited by I/O operations from the other adapter. Theoretically, each adapter card could drive its domains at 160 MB/s, giving 320 MB/s throughput on a single loop! The benefit of domains may reduce slightly over time, due to disk failures causing the groups to become intermixed, but the main benefits of spatial reuse will still apply.

If a cluster should fail, the remaining cluster device adapter will own all the domains on the loop, thus allowing full data access to continue.

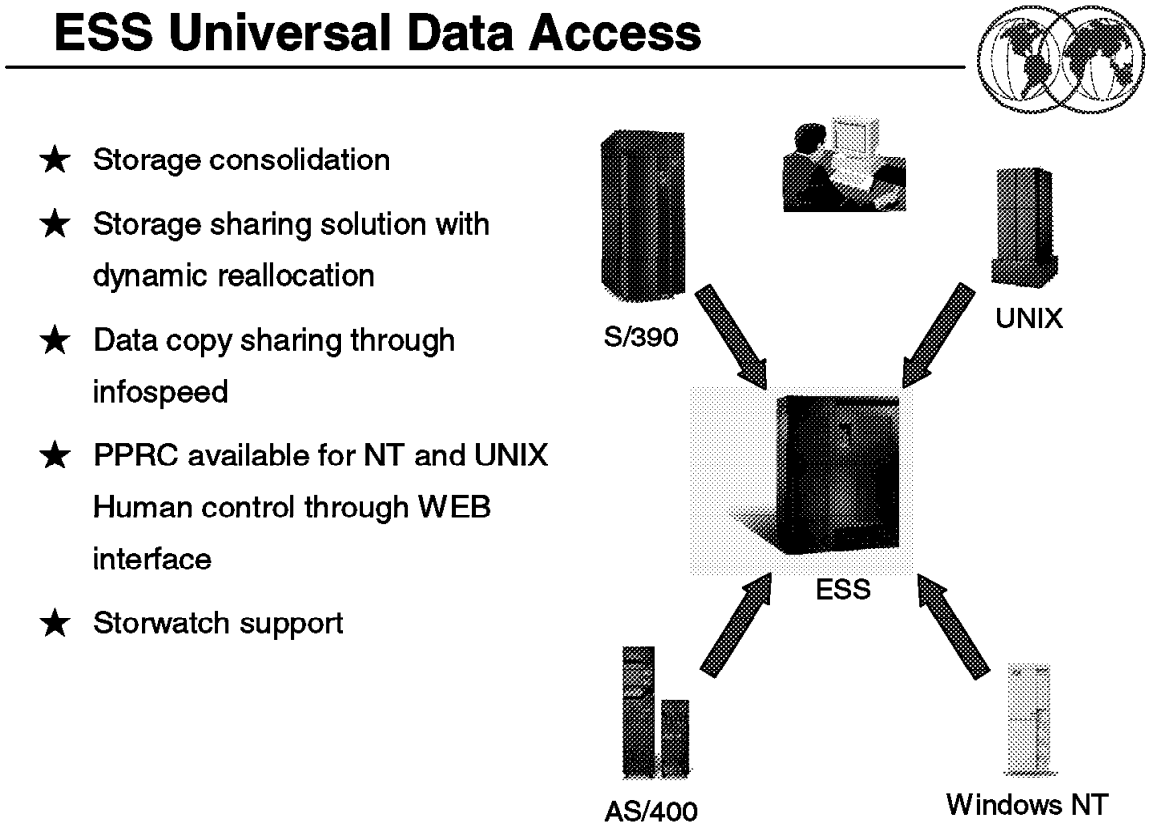


Figure 19. ESS universal access

1.8.4 ESS universal access

ESS is a product designed to implement storage consolidation that puts under the same cover all your enterprise data. This consolidation is the first step in achieving server consolidation, that is to put under the same MVS cluster all your enterprise applications.

Thinking in Seascape terms about universal access, the ESS box implements storage sharing with dynamic reallocation and data copy sharing using Infospeed.

Many of the ESS features are now available to non-S/390 platforms such as PPRC for NT and UNIX, where the control is through a Web interface

In the software side, there is StorWatch, a range of products in UNIX/NT that does what DFSMS and automation do for S/390.

In the next visual you will see all the operating systems able to access data in the ESS box in storage sharing mode (physical partition), that is, one device accessed by just one heterogeneous operating system image.

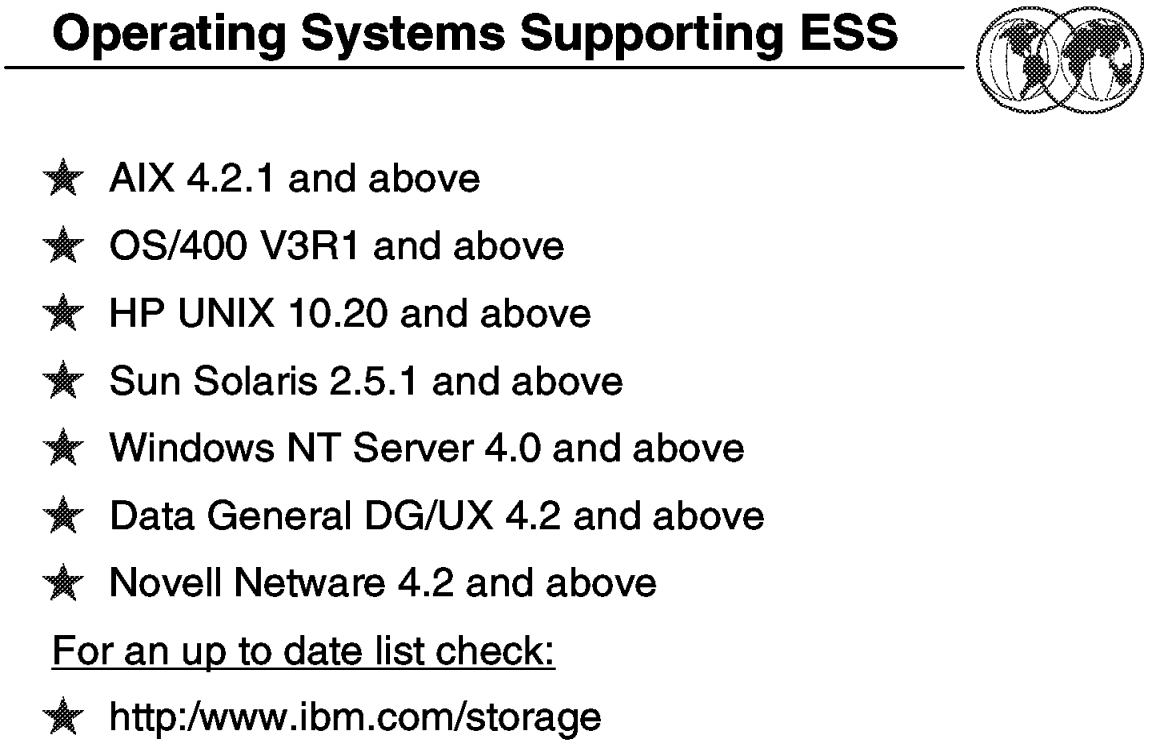


Figure 20. Operating systems supporting ESS

1.8.5 Operating systems supporting ESS

ESS is supported by the open operating systems shown in the visual. The list contains those systems able to access data in the ESS box at general availability time.

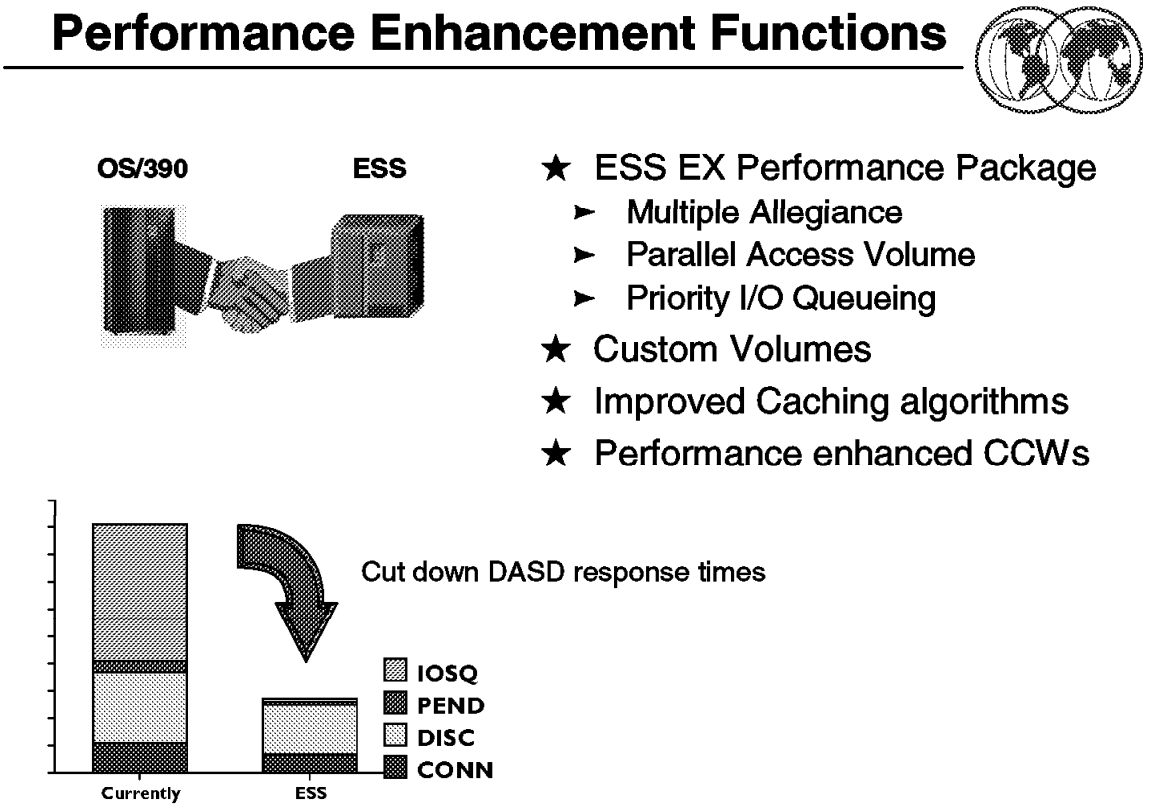


Figure 21. ESS new performance functions

1.8.6 ESS new performance functions

The new functions that ESS supports for S/390 (mainly OS/390) are:

- ESS EX performance package which consists of three performance features:
  - I/O priority queueing
  - Parallel Access Volumes (PAV), which is a chargeable feature
  - Multiple allegiance
- Performance enhanced channel command words (CCWs)
- Improved caching algorithms - good for cache friendly workloads, as the ones from OS/390
- Improved internal rates and parallelism - good for cache unfriendly workloads, as the ones from lower platforms. One example of parallelism is that RAID functions (16) are executed at disk loop level, not by the RISC server.
- Performance enhanced channel command words (CCWs)
- Custom volumes

1.8.6.1 I/O priority queueing

Previously to ESS, IOS kept the UCB I/O pending requests in a queue named IOSQ. The order in this queue - when the MVS image is in goal mode - is controlled by WLM. There was not this concept of priority queueing within the internal queues of the control units. With ESS, it is possible to have this queue concept internally. I/O Priority Queueing has the following properties:

- I/O can be queued with the ESS in priority order
- WLM sets the I/O priority when running in goal mode
- I/O priority for systems in a sysplex
- Fair share for each system

1.8.6.2 Parallel Access Volumes (PAV) and multiple allegiance

Traditional S/390 architecture does not allow more than one I/O operation to the same S/390 device, because such device only can handle one I/O operation at time.

However, in modern DASD subsystems as RAMAC and ESS, the S/390 device physically speaking is only a logical view. The contents of this logical device are spread in HDA RAID arrays and in caches. So, it is technically possible to have more than one I/O operation towards the same logical device.

Changes are made in MVS (in IOS code), in channel subsystem (SAP), and in ESS in order to allow more than one I/O operation on the same logical device. It is called parallel I/O which has two flavors:

- Parallel Access Volume (PAV), when the concurrent I/Os originate from the same MVS image
- Multiple Allegiance,

However, this concurrency can be achieved as long as no data accessed by one channel program can be altered through the actions of another channel program.

To implement PAV, the IOS introduces the concept of alias addresses. Instead of one UCB per logical volume, an MVS host can now use several UCBs for the same logical volume. Apart from the conventional Base UCB, alias UCBs can be defined and used by OS/390 to issue I/Os in parallel to the same logical volume device.

1.8.6.3 Performance enhanced channel command words (CCWs)

In ESS there is less overhead associated with CCW chains by combining tasks into fewer CCWs, introducing Read Track Data and Write Track Data CCWs. They allow reading and writing more data with fewer CCWs. It will be used by OS/390 to reduce ESCON protocol for multiple record transfer chains. Measurements on 4 KB records using an EXCP channel program showed a 15 percent reduction in channel overhead for the Read Track Data CCW.

1.8.6.4 Custom volumes

Custom volumes provides the possibility of defining small size 3390 or 3380 volumes. This causes less contention on a volume. Custom volumes is designed for high activity data sets. Careful size planning is required.

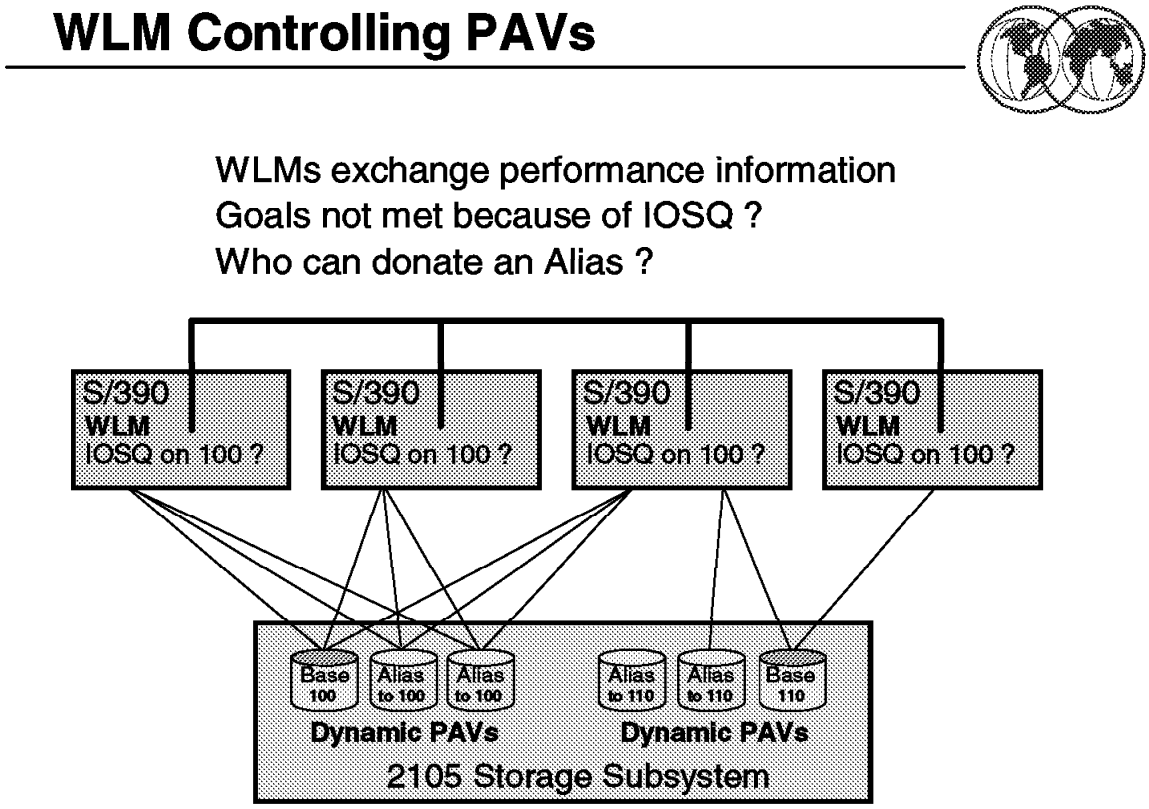


Figure 22. WLM controlling PAVs

1.8.7 WLM controlling PAVs

Through WLM, there are two mechanisms to tune the alias assignment:

- The first mechanism is goal based. This logic attempts to give additional aliases to a PAV-device that is experiencing IOS queue delays and is impacting a service class period that is missing its goal. To give additional aliases to the receiver device, a donor device must be found with a less important service class period. A bitmap is maintained with each PAV-device that indicates the service classes using the device.
- The second is to move aliases to high contention PAV-devices from low contention PAV-devices. High contention devices will be identified by having a significant amount of IOS queue time (IOSQ). This tuning is based on efficiency rather than directly helping a workload to meet its goal.

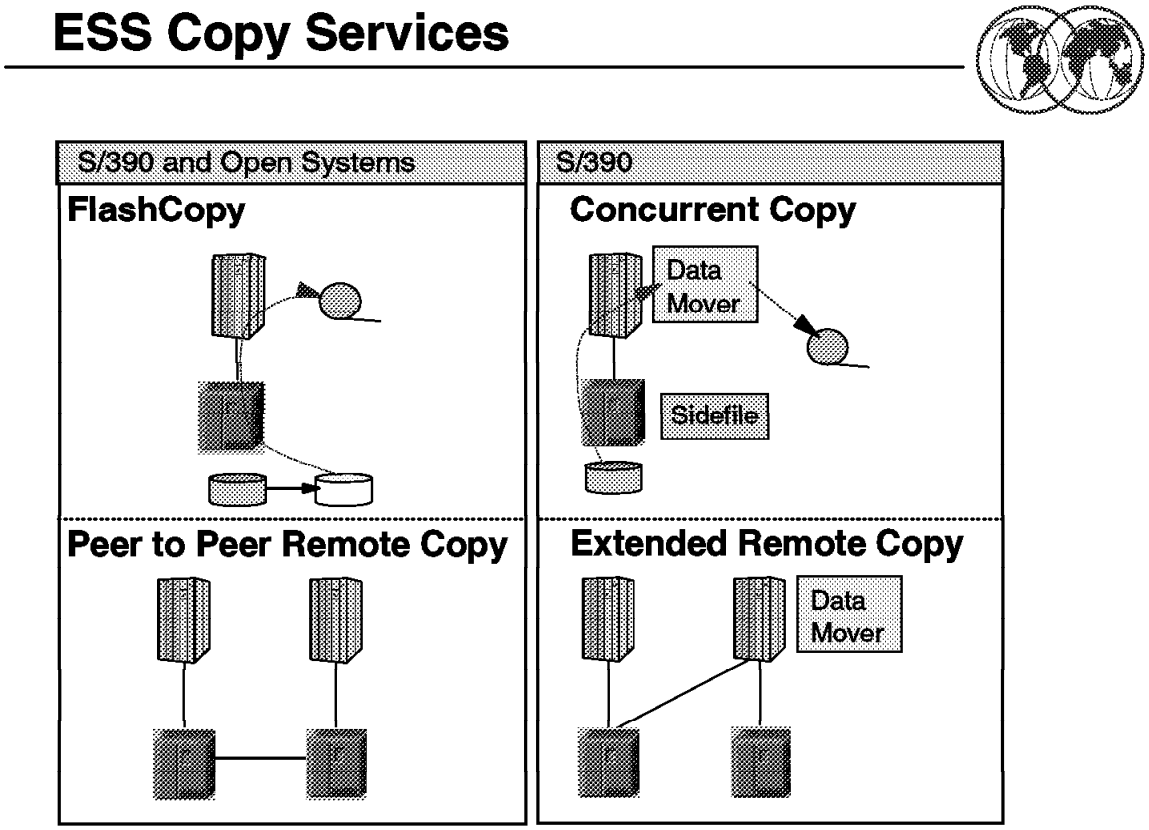


Figure 23. ESS copy services

1.8.8 ESS copy services

The following copy services are provided by the ESS:

- Enhanced PPRC service
- XRC suspend/resume service for unplanned outages
- Concurrent Copy
- FlashCopy service

1.8.8.1 Peer-to-peer remote copy (PPRC)

The peer-to-peer remote copy (PPRC) service is a hardware-based remote copy service that provides a synchronous volume copy across 3990 Model 6 storage subsystems. It is used for disaster recovery, device migration, and workload migration. For example, PPRC enables you to switch to a recovery system in the event of a disaster in an application system.

You can issue the CQUERY command to query the status of one volume of a PPRC volume pair or to collect information about a volume in the simplex state. The CQUERY command is modified and enabled to report on the status of S/390 attached CKD devices.

See *DFSMS/MVS Remote Copy Administrator's Guide and Reference*, SC35-0169, for further information on the PPRC service and the CQUERY command.

1.8.8.2 Extended remote copy (XRC)

The extended remote copy (XRC) service is a DFSMSdfp function that automatically sends copies of updated data to a remote recovery system with almost no impact to application system operations. DFSMS system data mover, a DFSMSdfp component, provides support for XRC.

You can implement XRC with one or two systems. Let's suppose that you have two systems: an application system at one location and a recovery system at another. With these two systems in place, XRC can automatically update your data on the remote disk storage subsystem as you make changes to it on your application system.

You can use the XRC suspend/resume service for planned outages. You can still use this standard XRC service on systems attached to the ESS if these systems are installed with the toleration or transparency support.

The introduction of the ESS enhances this XRC capability. Now you can use the XRC suspend/resume service to accommodate unplanned outages. With the enhanced XRC suspend/resume service, you do not have to terminate your current XRC sessions during an unplanned outage. Instead, you just "suspend" your existing XRC sessions and "restart" them. You can use the ESS copy service on systems with the exploitation support.

1.8.8.3 Concurrent Copy

Concurrent Copy allows for an instant T0 copy (S/390 only). It works on a volume or data set basis and uses cache side files in the ESS.

1.8.8.4 FlashCopy service

The FlashCopy service provides the appearance of instantaneous replication of a range of track images. This service requires both the source and target volumes to reside in a single logical subsystem. You can use the FlashCopy service to create copies for:

- Disaster recovery
- Business intelligence applications
- Data in a test environment
- Instantaneous checkpoints

The ESS FlashCopy service is compatible with the existing RAMAC Virtual Array (RVA) SnapShot capability provided by DFSMSdss. Therefore, you can invoke the FlashCopy service on the ESS with DFSMSdss.

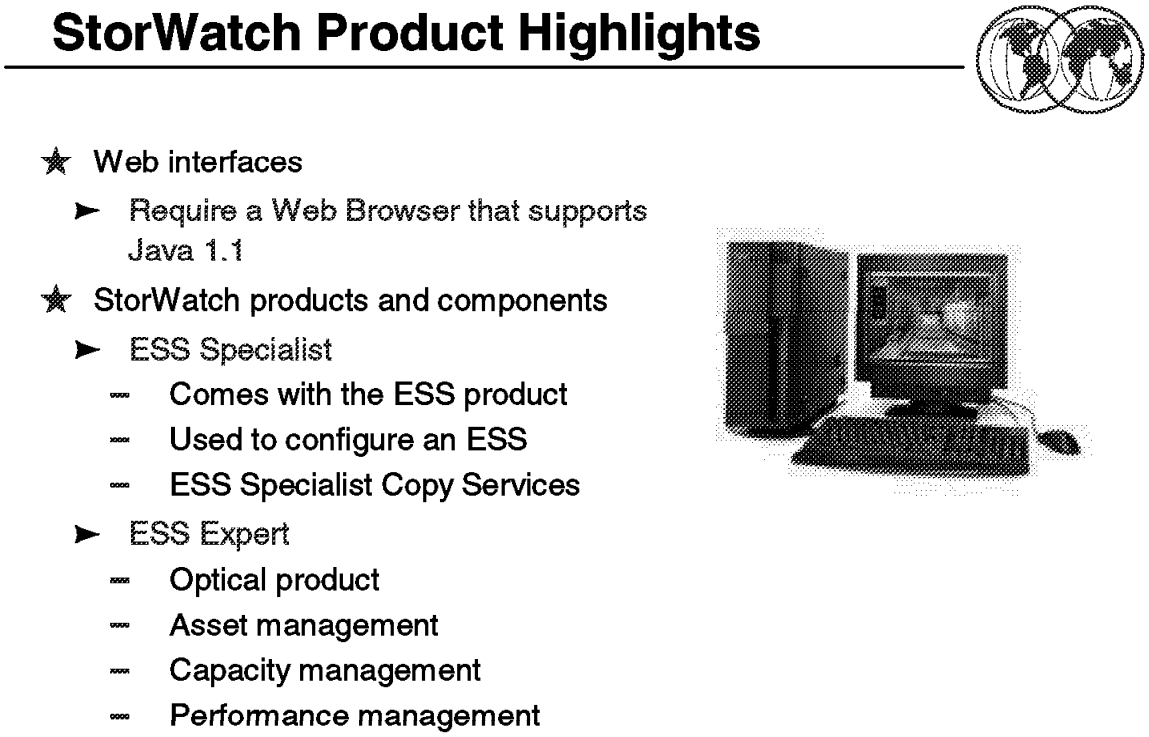


Figure 24. StorWatch product highlights

1.8.9 StorWatch product highlights

StorWatch, IBM′s Enterprise Storage Resource Management (ESRM) solution, is a growing software family whose goal is to enable storage administrators to efficiently manage storage resources from any location within an enterprise; enabling widely dispersed, disparate storage resources to be viewed and managed through a single, cohesive control point.

The product members of the StorWatch family are:

- StorWatch Reporter
- StorWatch Enterprise Storage Server Expert
- StorWatch Serial Storage Expert (StorX)
- StorWatch DFSMShsm Monitor Version 1 Release 1
- StorWatch Enterprise Storage Server Specialist

StorWatch products use a normal Web browser as the user interface. The only requirement is that the browser must support Java 1.1.

StorWatch ESS Specialist

The StorWatch ESS Specialist comes with the ESS product. The ESS Specialist must be used to configure an ESS. Apart from the configuration function for ESS the ESS Specialist can be used to administer Copy Services functions to set up FlashCopy T0 copies or Peer-to-Peer Remote Copy. Also there are functions for capacity planning and performance measurement.

Introduction to Tape Processing

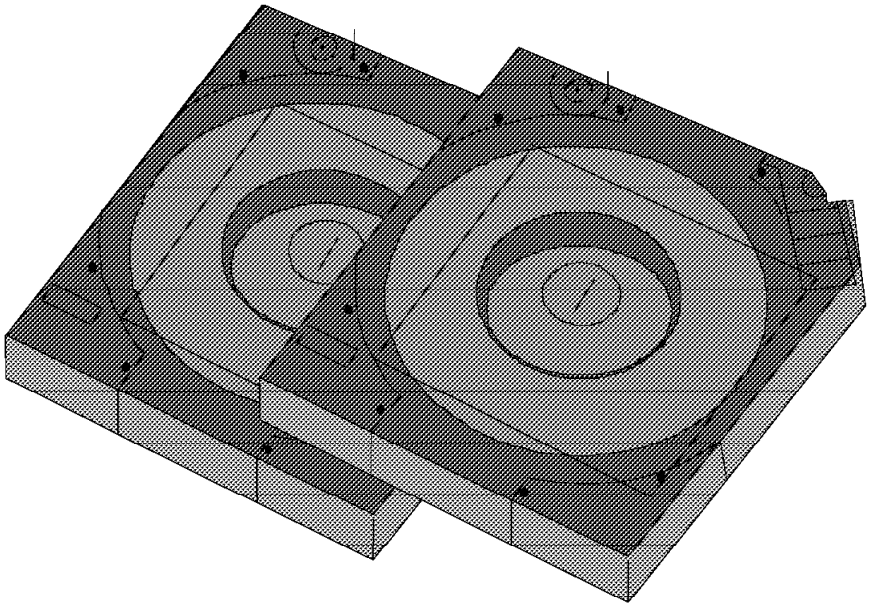


Figure 25. Introduction to tape processing

1.9 Introduction to tape processing

Tape are volumes that can be physically moved. You can store just sequential data sets on tape. Tape volumes can be sent to a safe or to other data processing centers.

Internal labels are used to identify magnetic tape volumes and the data sets on those volumes. You can process tape volumes with:

- IBM standard labels
- Labels that follow standards published by:
  - International Organization for Standardization (ISO)
  - American National Standards Institute (ANSI)
  - Federal Information Processing Standard (FIPS)
- Nonstandard labels
- No labels.

Your installation can install a bypass for any type of label processing; however, the use of labels is recommended as a basis for efficient control of your data.

IBM standard tape labels consist of volume labels and groups of data set labels. The volume label, identifying the volume and its owner, is the first record on the tape. The data set label, identifying the data set and describing its contents, precedes and follows each data set on the volume:

- The data set labels that precede the data set are called header labels.
- The data set labels that follow the data set are called trailer labels. They are almost identical to the header labels.
- The data set label groups can include standard user labels at your option.

Usually, the formats of ISO and ANSI labels, which are defined by the respective organizations, are similar to the formats of IBM standard labels.

Nonstandard tape labels can have any format and are processed by routines you provide. Unlabeled tapes contain only data sets and tapemarks.

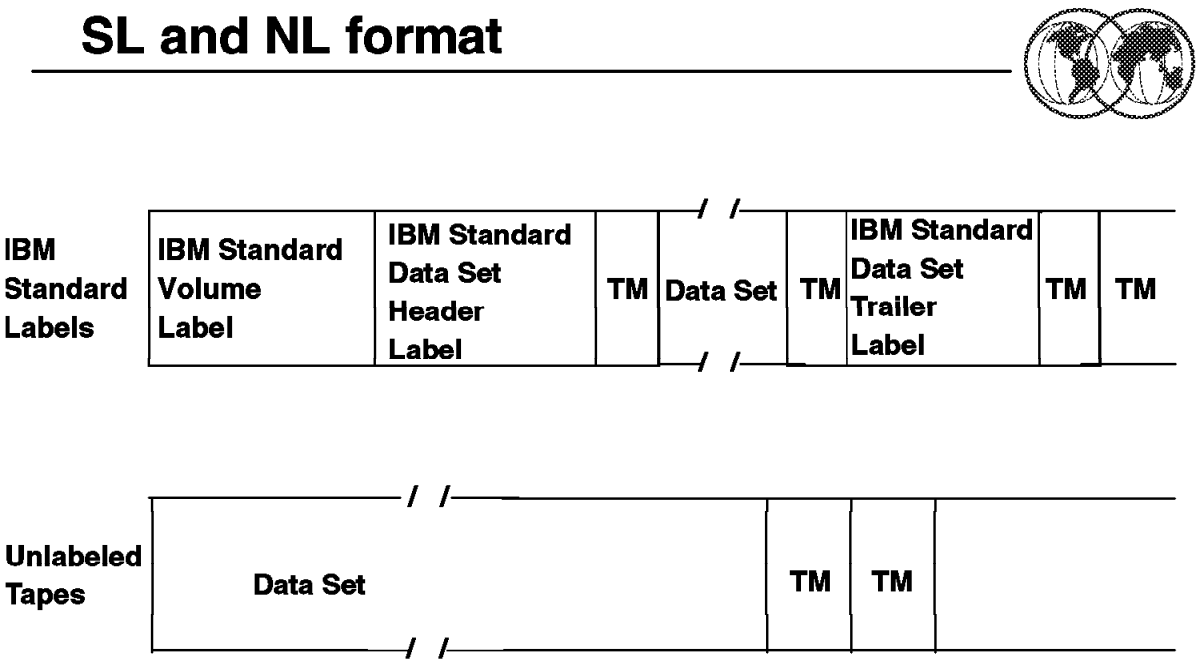


Figure 26. SL and NL

1.9.1 Describing the labels

In the job control statements, you must provide a data definition (DD) statement for each data set to be processed. The LABEL parameter of the DD statement is used to describe the data set's labels. You specify the type of labels by coding one of the following subparameters of the LABEL parameter as shown in table Table 3:

Table 3 (Page 1 of 2). Types of labels

Code	Meaning
SL	IBM Standard Label
AL	ISO/ANSI/FIPS labels
SUL	Both IBM Standard and user header or trailer labels
AUL	Both ISO/ANSI/FIPS and user header or trailer labels
NSL	Nonstandard labels
NL	No labels, but the existence of a previous label is verified
BLP	Bypass label processing. The data set is treated in the same manner as if NL had been specified, except that the system does not check for an existing volume label. The user is responsible for the positioning. If your installation does not allow BLP, the data set is treated exactly as if NL had been specified. Your job can use BLP only if Job Entry Subsystem (JES) through Job class, RACF through Tapevol class, or DFSMSrmm(*) allow it.

Table 3 (Page 2 of 2). Types of labels	
Code	**Meaning**
LTM	Bypass a leading tapemark, if encountered, on unlabeled tapes from VSE.

If you do not specify the label type, the operating system assumes that the data set has IBM standard labels.

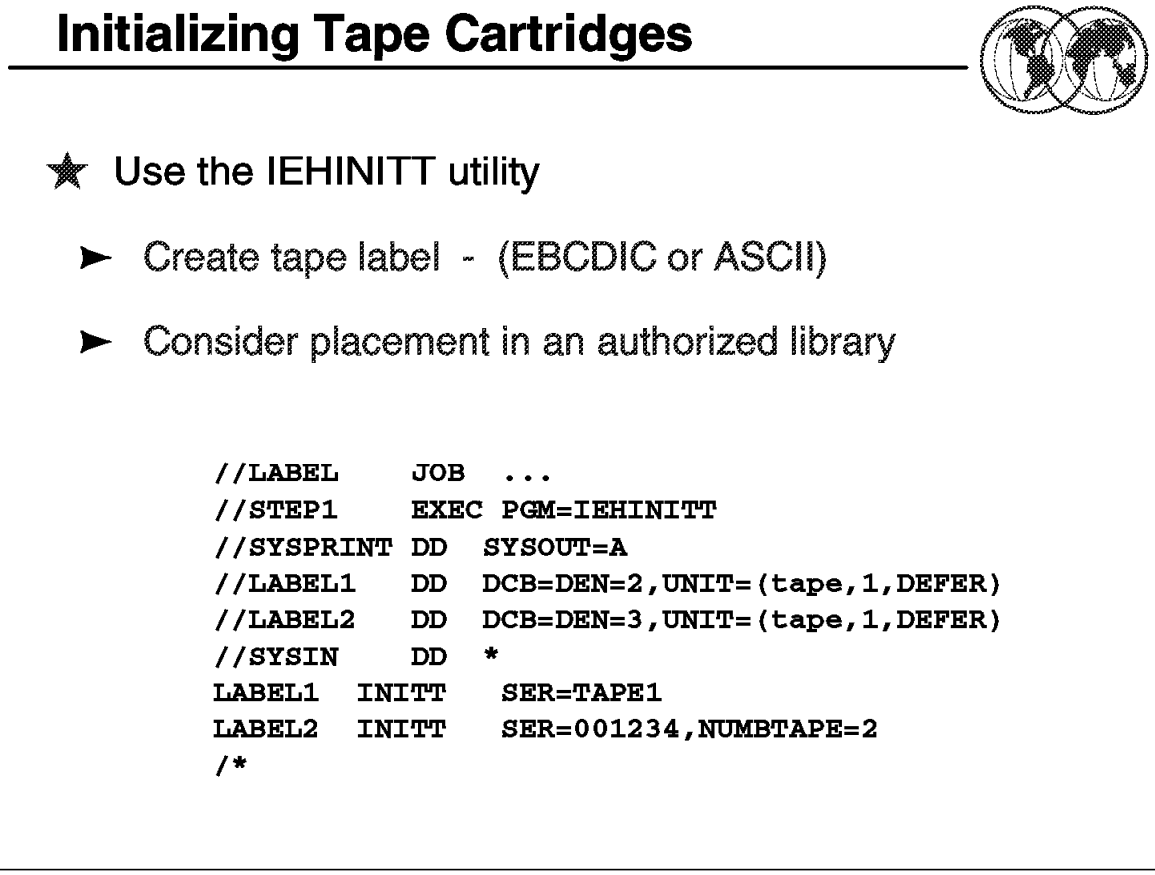


Figure 27. Initializing tape cartridges

1.9.2 Initializing tape cartridges

You may initialize tape volumes with two utilities:

- IEHINITT utility

  IEHINITT is a system utility used to place IBM volume label sets (no data set labels) written in EBCDIC (BCD for seven-track), or ISO/ANSI/FIPS volume label sets written in ASCII (American Standard Code for Information Interchange) onto any number of magnetic tapes mounted on one or more tape units.

  Because IEHINITT can overwrite previously labeled tapes regardless of expiration date and security protection, IEHINITT should be moved into an authorized password-protected private library and deleted from SYS1.LINKLIB. To further protect against overwriting the wrong tape, IEHINITT asks the operator to verify each tape mount.

  In the example, shown in Figure 28 on page 46, serial number TAPE1 is placed on a tape volume, and serial numbers 001234 and 001235 are placed on two tape volumes. The labels are written in EBCDIC at 800 and 1600 bits per inch respectively.

```
//LABEL    JOB  ...
//STEP1    EXEC PGM=IEHINITT
//SYSPRINT DD  SYSOUT=A
//LABEL1   DD  DCB=DEN=2,UNIT=(tape,1,DEFER)
//LABEL2   DD  DCB=DEN=3,UNIT=(tape,1,DEFER)
//SYSIN    DD  *
LABEL1  INITT   SER=TAPE1
LABEL2  INITT   SER=001234,NUMBTAPE=2
/*
```

Figure 28. IEHINITT example to write EBCDIC labels in different densities

In the example, shown in Figure 29, serial numbers 001234, 001244, 001254, 001264, 001274, and so forth, are placed on eight tape volumes. The labels are written in EBCDIC at 800 bits per inch. Each volume labeled is mounted, when it is required, on one of four nine-track tape units.

```
//LABEL4   JOB  ...
//STEP1    EXEC PGM=IEHINITT
//SYSPRINT DD  SYSOUT=A
//LABEL    DD  DCB=DEN=2,UNIT=(tape,4,DEFER)
//SYSIN    DD  *
LABEL   INITT   SER=001234
LABEL   INITT   SER=001244
LABEL   INITT   SER=001254
LABEL   INITT   SER=001264
LABEL   INITT   SER=001274
LABEL   INITT   SER=001284
LABEL   INITT   SER=001294
LABEL   INITT   SER=001304
/*
```

Figure 29. Place serial number on eight tape volumes

- EDGINERS utility

  The EDGINERS utility program verifies that the volume is mounted before writing a volume label on a labeled, unlabeled, or blank tape. EDGINERS does not check password or RACF security protection, but it verifies whether the volume is defined to DFSMSrmm. DFSMSrmm must know that the volume needs to be labelled. If the labelled volume is undefined, then DFSMSrmm defines it to DFSMSrmm and can create RACF volume security protection.

  Detailed procedures for using the program are described in *DFSMS/MVS DFSMSrmm Implementation and Customization Guide*, SC26-4932.

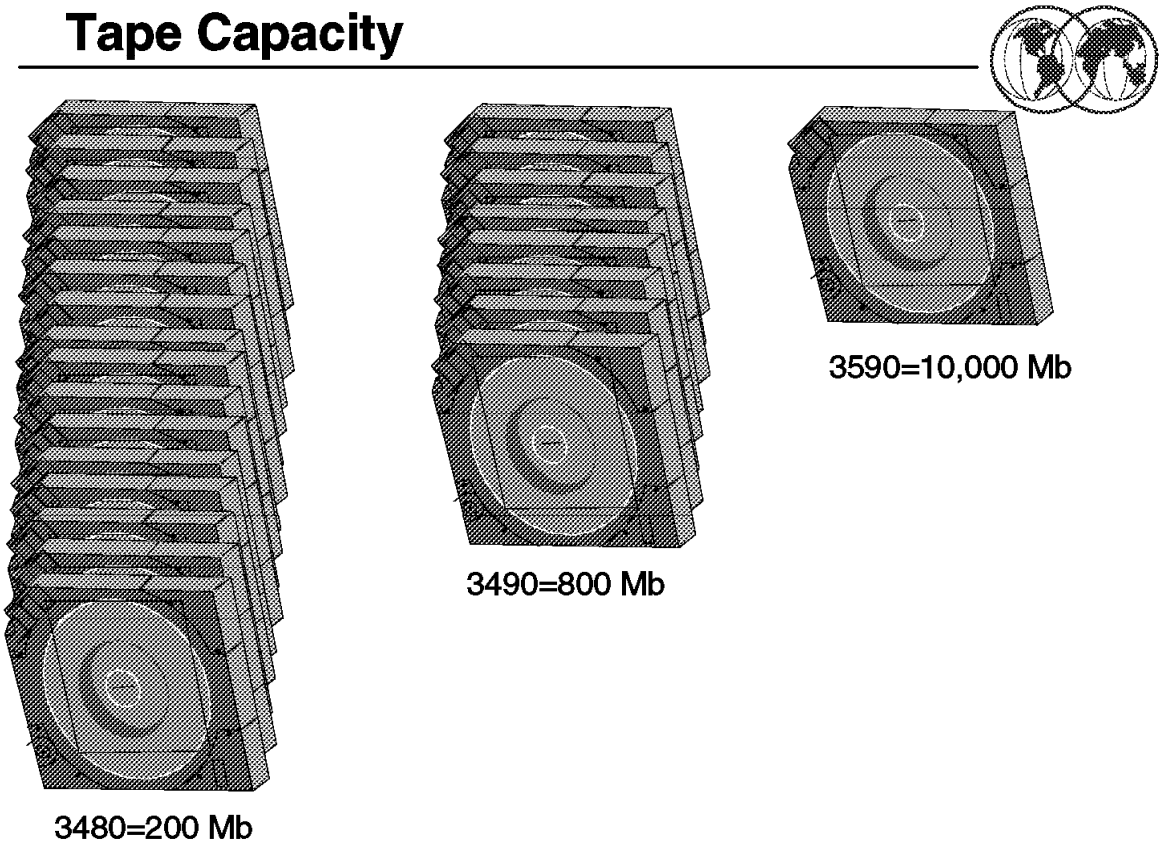


Figure 30. Tape capacity

1.9.3 Tape capacity

The capacity of a tape depends on the device type that is recording it. 3480 and 3490 tapes are physically the same cartridges. The IBM 3590 high performance cartridge tape is not compatible with the 3480, 3490, or 3490E drives. 3490 units can read 3480 cartridges, but cannot record as a 3480, and 3480 units cannot read or write as a 3490. Table 4 lists all IBM tape capacities supported since 1952.

Table 4. Tape capacity of various IBM products

Year	Product	Capacity (Mb)	Transfer Rate (KB/S)
1952	IBM 726	1.4	7.5
1953	IBM 727	5.8	15
1957	IBM 729	23	90
1965	IBM 2401	46	180
1968	IBM 2420	46	320
1973	IBM 3420	180	1250
1984	IBM 3480	200	3000
1989	IBM 3490	200	4500
1991	IBM 3490E	400	9000
1992	IBM3490E	800	9000
1995	IBM 3590 Magstar	10000 (uncompacted)	20000 (uncompacted)

-

For further information about tape processing, see *DFSMS/MVS Using Magnetic Tapes*, SC26-4923.

3494 Tape Library

Figure 31. 3494 tape library

1.9.4 3494 tape library

Tape storage media can provide low-cost data storage for sequential files, inactive data, and vital records. Because of the continued growth in tape use, *tape automation* has been seen as a way of addressing an increasing number of challenges. Various solutions that provide tape automation are available, including:

- The Automatic Cartridge Loader on IBM 3480 and 3490E tape subsystems, which provides quick scratch (a volume with no valued data, used for output) mount.
- The Automated Cartridge Facility on the Magstar 3590 tape subsystem, which, working with application software, can provide a 10-cartridge mini-tape library.
- The IBM 3494 (an automated tape library dataserver) which can provide up to 62 tape drives and store up to 187 TB of compacted data. An automated tape library dataserver is a device consisting of robotics components, cartridge storage areas (or shelves), tape subsystems, and controlling hardware and software, together with the set of tape volumes that reside in the library and can be mounted on the library tape drives.
- The Magstar Virtual Tape Server (VTS), which provides “volume stacking” capability and exploits the capacity and bandwidth of Magstar 3590 technology.

IBM 3494 offers a wide range of models and features:

- Up to 62 tape drives.

- Support through the Library Control Unit for attachment of up to 15 additional frames including the Magstar VTS, for a total of 16 frames, not including the High Availability unit.
- Cartridge storage capacity of 210 to 6240 tape cartridges.
- Data storage capacity of up to 62.4 TB of uncompacted data and 187 TB of compacted data.
- Support of the High Availability unit that provides a high level of availability for tape automation.
- Support of the Magstar VTS.
- Support for the IBM 3490E Model F1A tape drive, IBM 3490E Model CxA tape drives, IBM Magstar 3590 Model B1A tape drive, and IBM Magstar 3590 Model A00 or A50 tape controller.
- Attachment to and sharing by multiple host systems, such as S/390, RS/6000, AS/400, HP, and Sun processors.
- Data paths through SCSI-2, ESCON, and parallel channels depending on the tape subsystem installed.
- Library management commands through RS-232, a local area network (LAN), and parallel and ESCON channels.

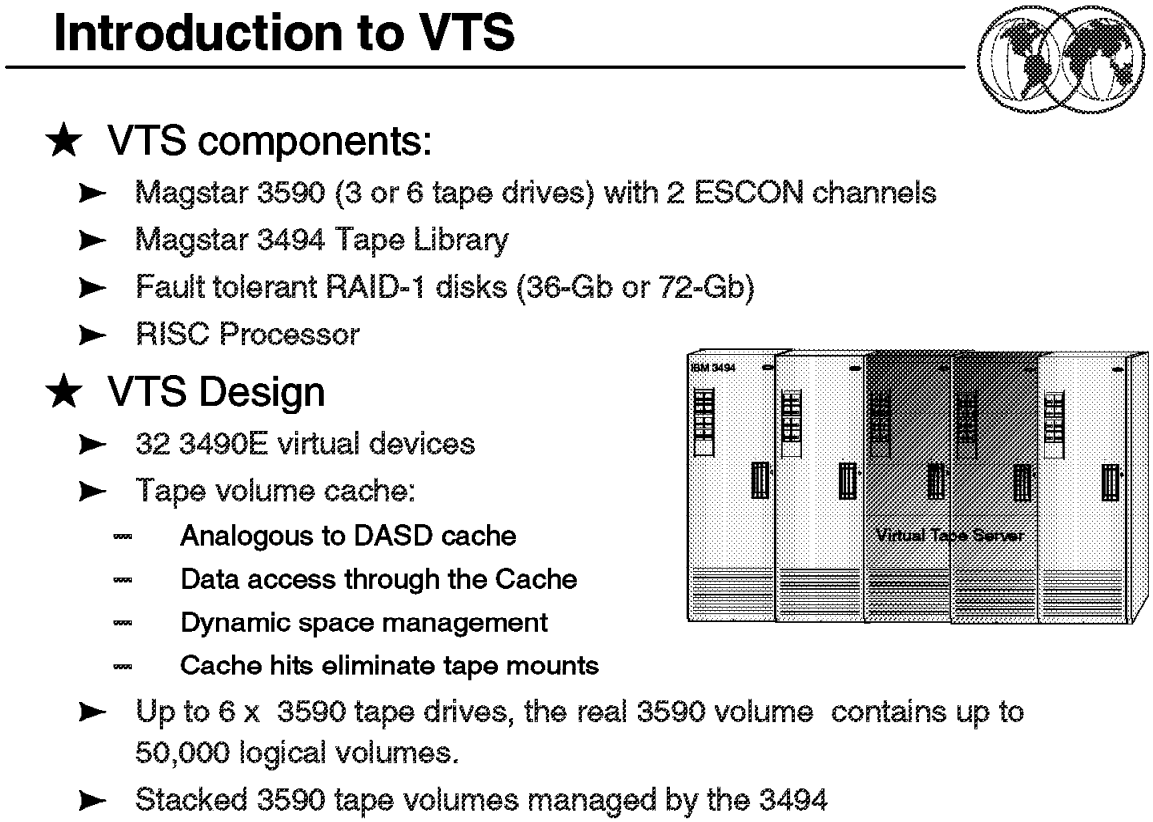


Figure 32. Introduction to VTS

1.10 Introduction to VTS

The IBM Magstar Virtual Tape Server (VTS), integrated with the IBM Tape Library Dataservers (3494), delivers an increased level of storage capable to the traditional storage products hierarchy. The host software sees VTS as a 3490 Enhanced Capability (3490E) Tape Subsystem with associated standard (CST) or Enhanced Capacity Cartridge System Tapes (ECCST). This virtualization of both the tape devices and the storage media to the host allows for transparent utilization of the capabilities of the IBM 3590 tape technology.

Along with introducing the IBM Magstar VTS, IBM introduced new views of volumes and devices because of the different knowledge about volumes and devices in the host system and the hardware. Using a VTS subsystem, the host application writes tape data to virtual devices. The volumes created by the hosts are called Virtual Volumes and are physically stored in a tape volume cache which is built from RAID DASD.

Through tape volume cache management policies, the VTS management software moves host-created volumes from the tape volume cache to a Magstar cartridge managed by the VTS subsystem. When a virtual volume is moved from the tape volume cache to tape, it becomes a logical volume.

VTS looks like an automatic tape library with 32 3490E drives, 50000 volumes in 37 square feet. Its major components are:

- Magstar 3590 (three or six tape drives) with two ESCON channels
- Magstar 3494 Tape Library

- Fault tolerant RAID-1 disks (36-Gb or 72-Gb)
- RISC Processor

VTS provides the following functions:

- 32 3490E virtual devices.
- Tape volume cache (implemented in a RAID-1 disk), it contains virtual volumes. At close time the virtual volume is copied to logical volumes in the 3590 tape volumes:
  - Analogous to DASD cache
  - Data access through the Cache
  - Dynamic space management
  - Cache hits eliminate tape mounts
- Up to six 3590 tape drives, the real 3590 volume contains logical volumes. Installation sees up to 50,000 volumes.
- Stacked 3590 tape volumes managed by the 3494.

VTS is expected to provide a ratio of 59:1 in volume reduction with dramatic savings in all tape hardware (drives, controllers, and robots).

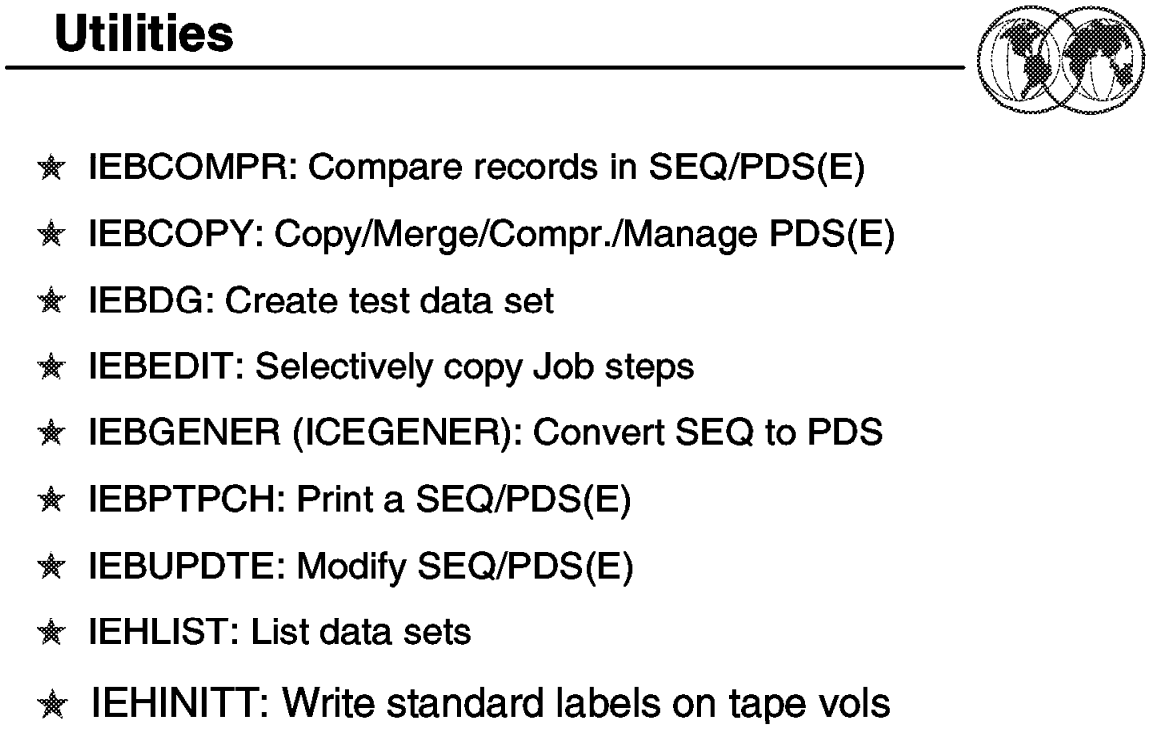


Figure 33. Utilities

1.11 Introduction to utilities

DFSMS/MVS provides utility programs to assist you in organizing and maintaining data. Utilities are programs which perform commonly needed functions. See “Guide to Utility Program Functions” in topic 1.1 of *DFSMS/MVS Utilities*, SC26-4926, to help you find the program that performs the function you need.

1.11.1 System utilities programs

System utility programs are used to list or change information related to data sets and volumes, such as data set names, catalog entries, and volume labels. Most functions that system utility programs can perform are performed more efficiently with other programs, such as IDCAMS, ISMF, or DFSMSrmm. See Table 5 for a description of system utilities. The ones logically replaced are marked with an asterisk (*):

Table 5 (Page 1 of 2). System utility programs

System utility	Alternate program	Purpose
*IEHINITT	DFSMSrmm EDGINERS	Write standard labels on tape volumes.
IEHLIST	ISMF, PDF 3.4	List system control data.
*IEHMOVE	DFSMSdss, IEBCOPY	Move or copy collections of data.

Table 5 (Page 2 of 2). System utility programs		
System utility	**Alternate program**	**Purpose**
IEHPROGM	PDF 3.2, Access Method Services	Build and maintain system control data.
*IFHSTATR	DFSMSrmm, EREP	Select, format, and write information about tape errors from the IFASMFDP tape.

1.11.2 Data set utility programs

You can use data set utility programs to reorganize, change, or compare data at the data set or record level. These programs are controlled by JCL statements and utility control statements.

These utilities allow you to manipulate partitioned, sequential or indexed sequential data sets, or partitioned data sets extended (PDSEs), which are provided as input to the programs. You can manipulate data ranging from fields within a logical record to entire data sets.

The data set utilities included in this topic cannot be used with VSAM data sets. Information about VSAM data sets can be found in *DFSMS/MVS Using Data Sets*, SC26-4922, and in a later topic 1.15, “Access method services” on page 70.

Table 6 is a list of data set utility programs and their use.

Table 6. Data set utility programs	
Data set utility	**Use**
*IEBCOMPR, SuperC, (PDF 3.12)	Compare records in sequential or partitioned data sets, or PDSEs.
IEBCOPY	Copy, compress, or merge partitioned data sets or PDSEs; add RLD count information to load modules; select or exclude specified members in a copy operation; rename or replace selected members of partitioned data sets or PDSEs.
IEBDG	Create a test data set consisting of patterned data.
IEBEDIT	Selectively copy job steps and their associated JOB statements.
IEBGENER or ICEGENER	Copy records from a sequential data set or convert a data set from sequential organization to partitioned organization.
*IEBIMAGE or AMS REPRO	Modify, print, or link modules for use with the IBM 3800 Printing Subsystem, the IBM 3262 Model 5, or the 4248 printer.
*IEBISAM	Unload, load, copy, or print an ISAM data set.
IEBPTPCH or PDF 3.1 or 3.6	Print or punch records in a sequential or partitioned data set.
IEBUPDTE	Incorporate changes to sequential or partitioned data sets, or PDSEs.

System and data set utility programs are controlled by job control and utility control statements. The job control and utility control statements necessary to use utility programs are provided in the major discussion of each utility program; for more information about control statements refer to *DFSMS/MVS Utilities*, SC26-4926. The next visuals show a few examples of using utility programs.

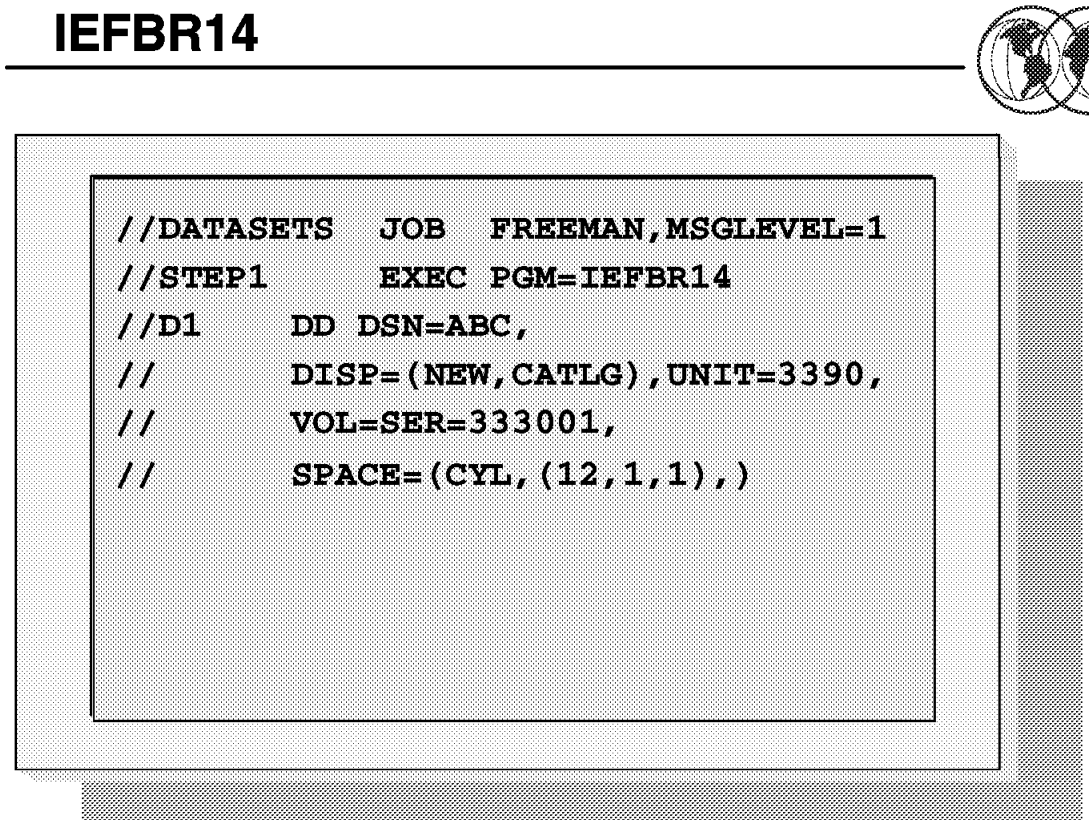


Figure 34. IEFBR14

1.11.3 IEFBR14

IEFBR14 is not a utility program, it is a two-line program that clears register 15, thus passing a return code of 0, and then branches to the address in register 14, which returns control to the system. So, in other words this program is dummy program. It can be used in a step to force MVS (specifically, the initiator) to process the JCL code and execute some functions as:

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

Considerations when using IEFBR14

Although the system allocates space for data sets, it does not initialize the new data sets. Therefore, any attempt to read from one of these new data sets in a subsequent step may produce unpredictable results. Also, IBM does not recommend allocation of multi-volume data sets while executing IEFBR14.

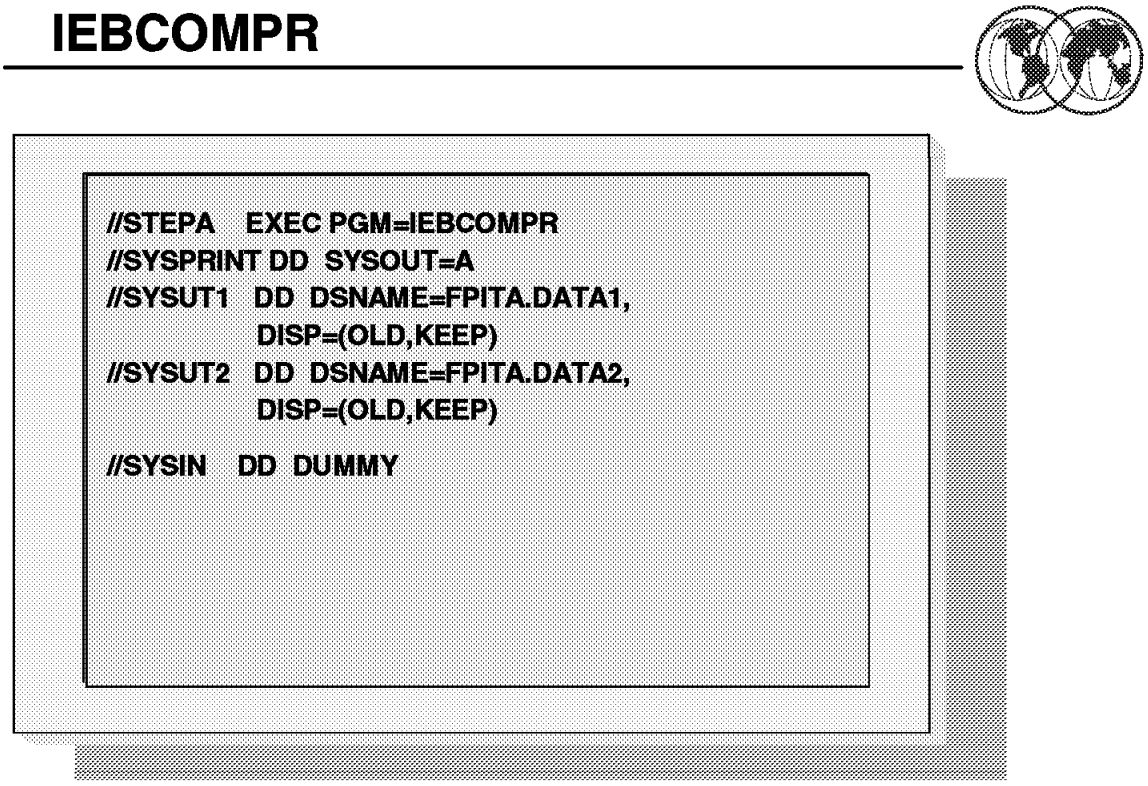


Figure 35. IEBCOMPR

1.11.4 IEBCOMPR (compare data set) program

IEBCOMPR is a data set utility used to compare two sequential data sets, two partitioned data sets, or two PDSEs at the logical record level to verify a backup copy. Fixed, variable, or undefined records from blocked or unblocked data sets or members can also be compared. However, you should not use IEBCOMPR to compare load modules.

Two sequential data sets are considered equal, that is, are considered to be identical, if:

- The data sets contain the same number of records, and
- Corresponding records and keys are identical

Two partitioned data sets or two PDSEs are considered equal if:

- Corresponding members contain the same number of records
- Note lists are in the same position within corresponding members
- Corresponding records and keys are identical
- Corresponding directory user data fields are identical

If all these conditions are not met for a specific type of data set, those data sets are considered unequal. If records are unequal, the record and block numbers, the names of the DD statements that define the data sets, and the unequal records are listed in a message data set. Ten successive unequal comparisons stop the job step, unless you provide a routine for handling error conditions.

Load module partitioned data sets that reside on different types of devices should not be compared. Under most circumstances, the data sets will not compare as equal.

Partitioned data sets or PDSEs can be compared only if all the names in one or both of the directories have counterpart entries in the other directory. The comparison is made on members identified by these entries and corresponding user data.

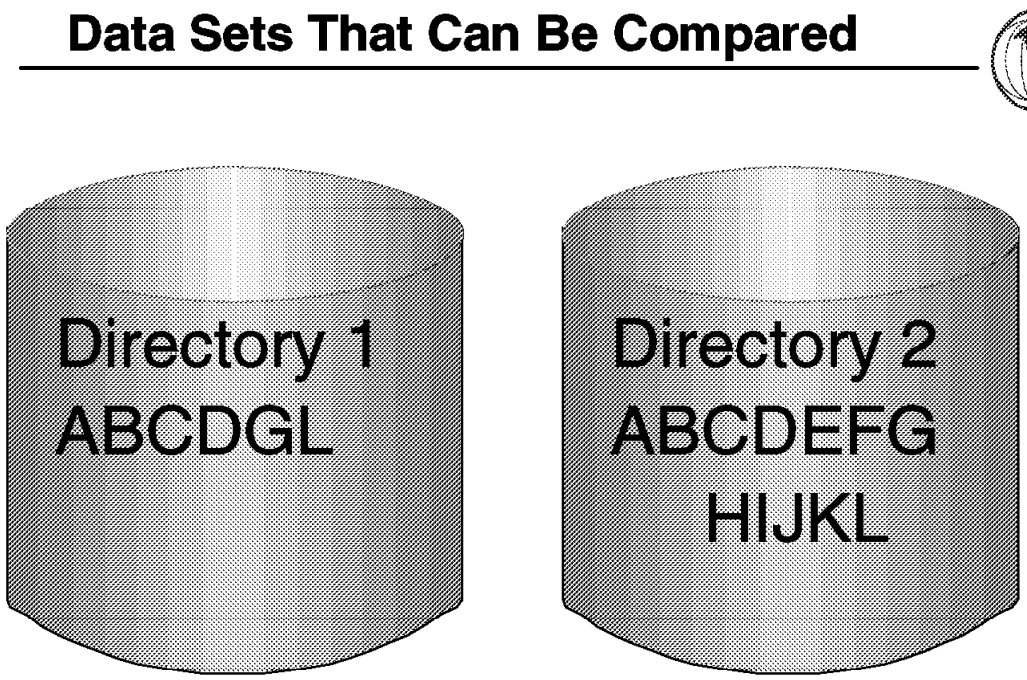


Figure 36. Directories of PO that can be compared

1.11.5 Data sets that can be compared

This visual shows the directories of two partitioned data sets. Directory 2 contains corresponding entries for all the names in Directory 1; therefore, the data sets can be compared.

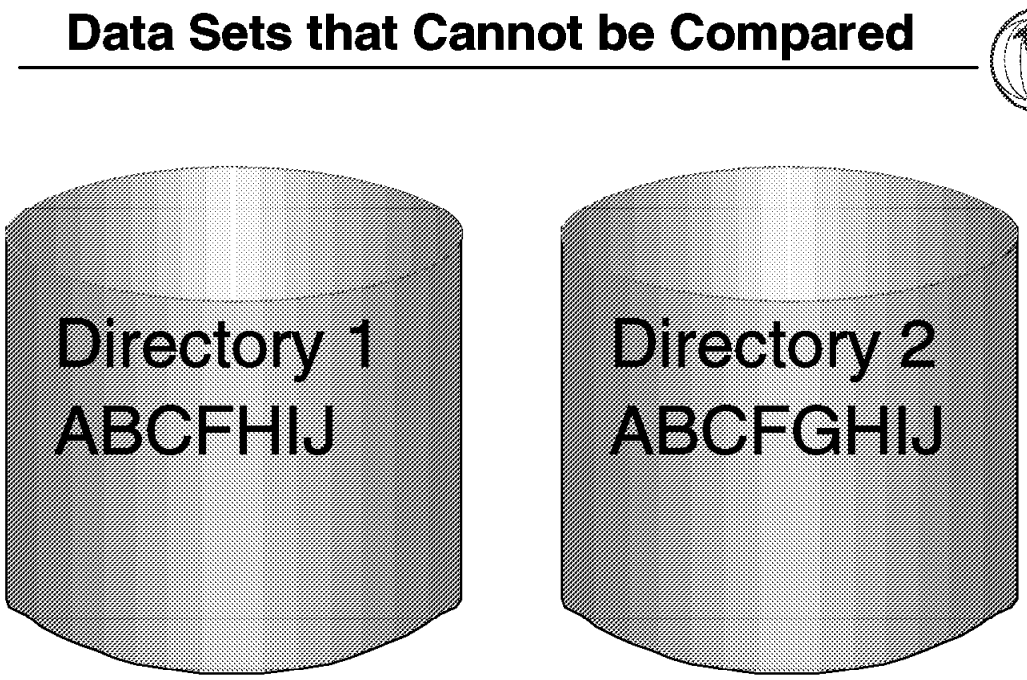


Figure 37. Directories of PO that cannot be compared

1.11.6 Data sets that cannot be compared

This visual shows the directories of two partitioned data sets. Each directory contains a name that has no corresponding entry in the other directory; therefore, the data sets cannot be compared, and the job step will be ended.

User exits are provided for optional user routines to process user labels handle error conditions, and modify source records. See *Appendix C, "Specifying User Exits with Utility Programs"* topic C.0 of *DFSMS/MVS Utilities*, SC26-4926, for a discussion of the linkage conventions to be followed when user routines are used.

IEBCOPY

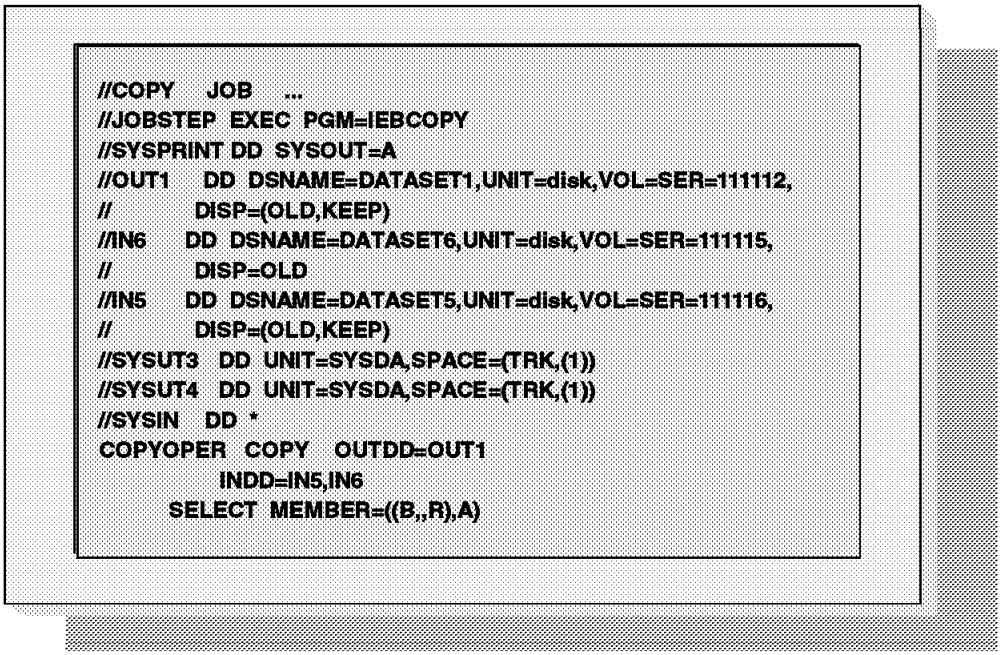


Figure 38. IEBCOPY

1.12 IEBCOPY

IEBCOPY is a data set utility used to copy or merge members between one or more partitioned data sets, or partitioned data sets extended (PDSE), in full or in part. You can also use IEBCOPY to create a backup of a partitioned data set into a sequential data set (called an unload data set or PDSU), and to copy members from the backup into a partitioned data set.

IEBCOPY is used to:

- Make a copy of a partitioned data set or PDSE
- Merge partitioned data sets (except when unloading)
- Create a sequential form of a partitioned data set or PDSE for a back up or transport
- Reload one or more members from a PDSU into a partitioned data set or PDSE
- Select specific members of a partitioned data set or PDSE to be copied, loaded, or unloaded
- Replace members of a partitioned data set or PDSE
- Rename selected members of a partitioned data set or PDSE
- Exclude members from a data set to be copied, unloaded, or loaded (except on COPYGRP)
- Compress a partitioned data set in place
- Upgrade an OS format load module for faster loading by MVS program fetch
- Copy and reblock load modules

- Convert load modules in a partitioned data set to program objects in a PDSE when copying a partitioned data set to a PDSE
- Convert a partitioned data set to a PDSE or a PDSE to a partitioned data set
- Copy to or from a PDSE data set, a member and its aliases together as a group (COPYGRP)

In addition, IEBCOPY automatically lists the number of unused directory blocks and the number of unused tracks available for member records in the output partitioned data set.

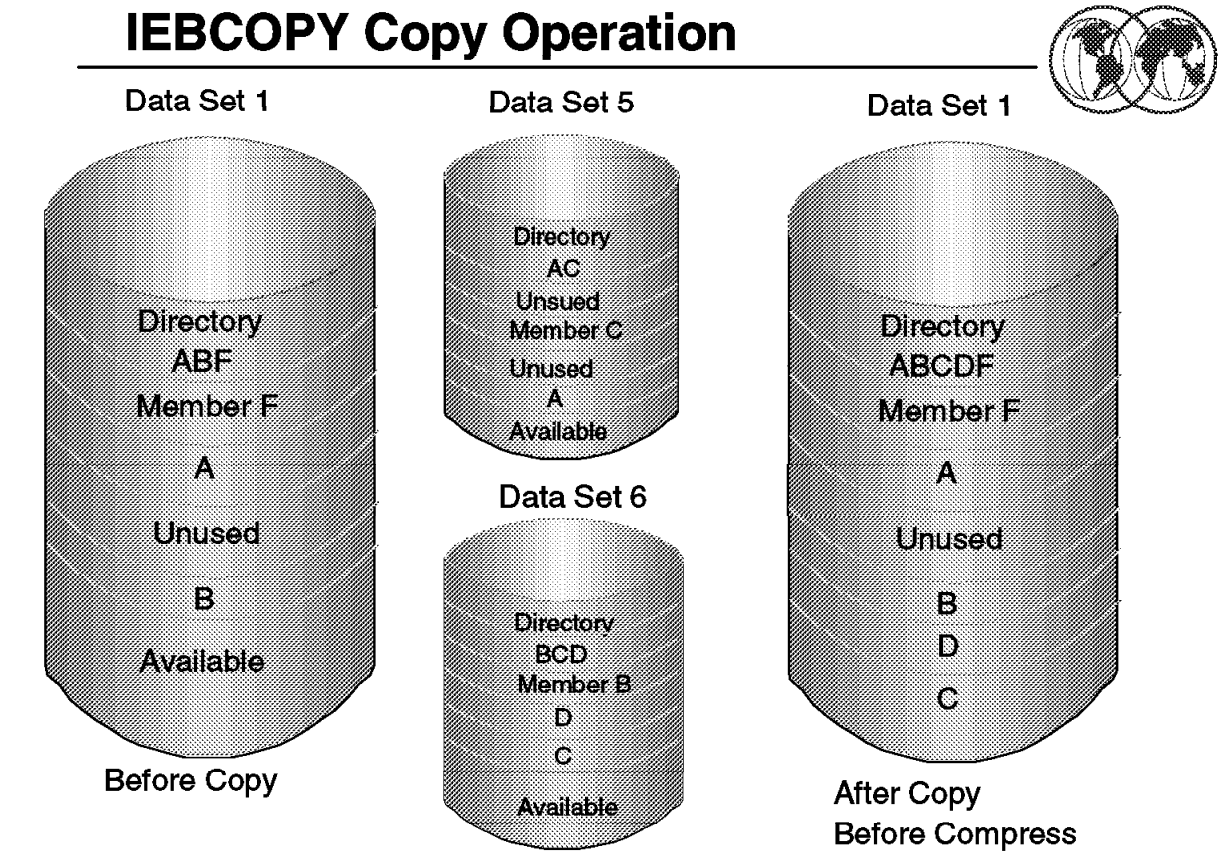


Figure 39. IEBCOPY copy operation

1.12.1 IEBCOPY copy operation

In this example, two input partitioned data sets (data set5 and data set6) are copied to an existing output partitioned data set (data set1). In addition, all members on data set6 are copied; members on the output data set that have the same names as the copied members are replaced. After data set6 is processed, the output data set (data set1) is compressed in place. The visual shows the input and output data sets before and after copy processing. The compress process will be shown in Figure 40 on page 64.

Following is the JOB that is used to copy and compress partitioned data sets:

```
//COPY     JOB  ...
//JOBSTEP  EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=A
//INOUT1   DD  DSNAME=data set1,UNIT=disk,VOL=SER=111112,
//             DISP=(OLD,KEEP)
//IN5      DD  DSNAME=data set5,UNIT=disk,VOL=SER=111114,
//             DISP=OLD
//IN6      DD  DSNAME=data set6,UNIT=disk,VOL=SER=111115,
//             DISP=(OLD,KEEP)
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD  *
COPYOPER COPY  OUTDD=INOUT1,INDD=(IN5,(IN6,R),INOUT1)
/*
```

The control statment is discussed below:

- INOUT1 DD defines a partitioned data set (data set1), which contains three members (A, B, and F).
- IN5 DD defines a partitioned data set (data set5), which contains two members (A and C).
- IN6 DD defines a partitioned data set (data set6), which contains three members (B, C, and D).
- SYSUT3 and SYSUT4 DD define temporary spill data sets. One track is allocated for each on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement.
- COPY indicates the start of the copy operation. The OUTDD operand specifies data set1 as the output data set.

Processing occurs as follows:

1. Member A is not copied from data set5 into data set1 because it already exists on data set1 and the replace option was not specified for data set5.
2. Member C is copied from data set5 to data set1, occupying the first available space.
3. All members are copied from data set6 to data set1, immediately following the last member. Members B and C are copied even though the output data set already contains members with the same names because the replace option is specified on the data set level.

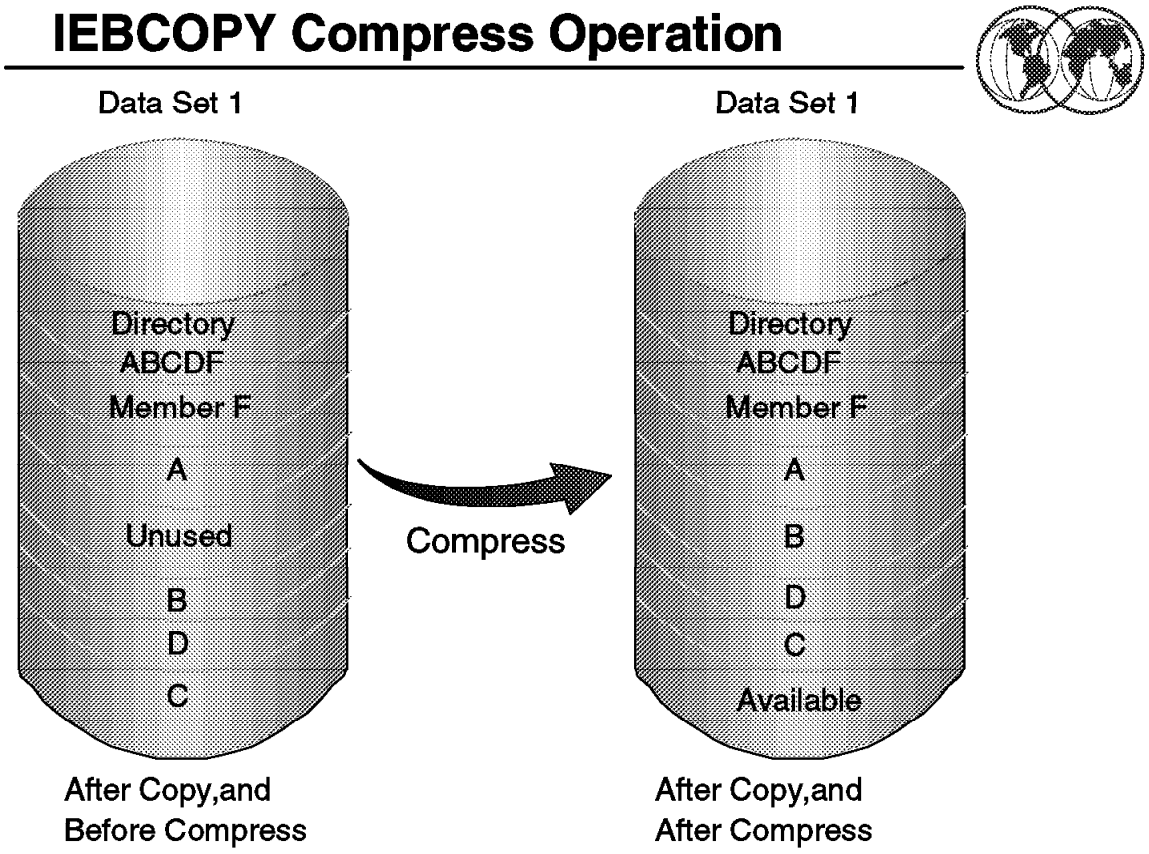


Figure 40. IEBCOPY compress operation

1.12.2 IEBCOPY compress operation

The pointers in data set1′s directory are changed to point to the new members B and C. Thus, the space occupied by the old members B and C is unused. The members currently on data set1 are compressed in place, thereby eliminating embedded unused space.

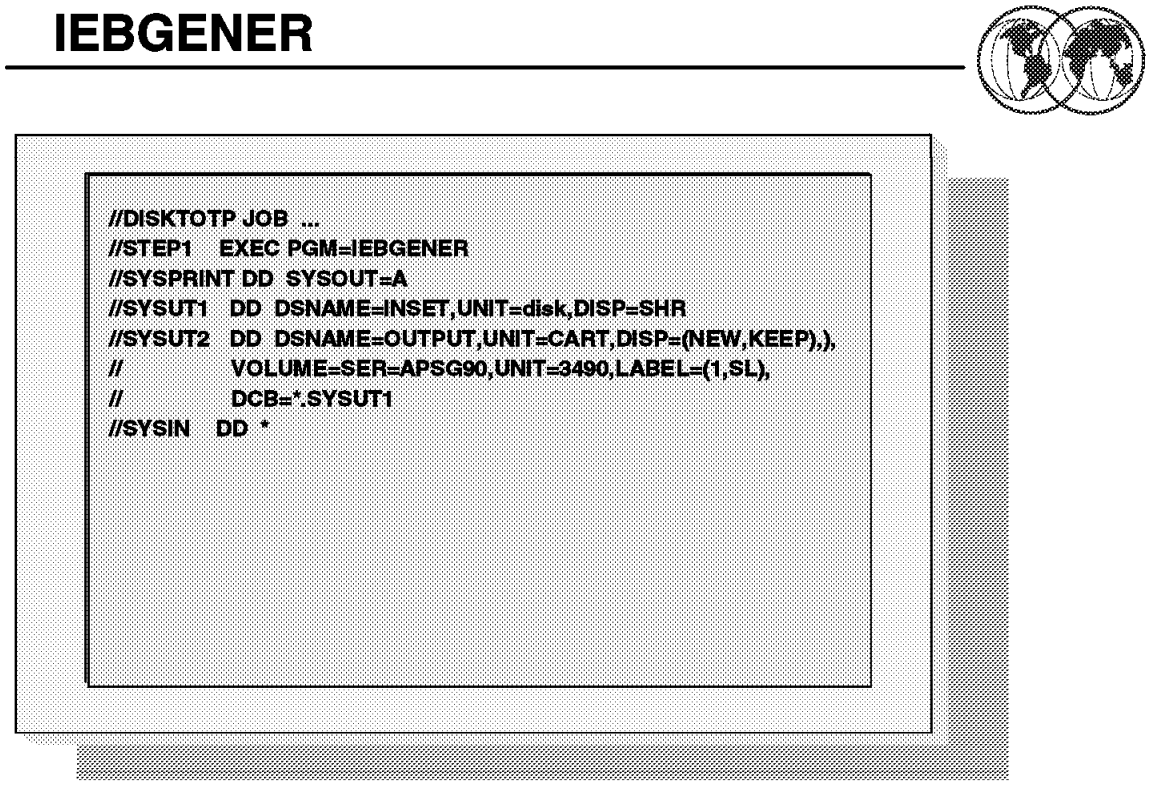


Figure 41. IEBGENER

1.13 IEBGENER

You can use IEBGENER to:

- Create a backup copy of a sequential data set or a member of a partitioned data set or PDSE.
- Produce a partitioned data set or PDSE, or a member of a partitioned data set or PDSE, from a sequential data set.
- Expand an existing partitioned data set or PDSE by creating partitioned members and merging them into the existing data set.
- Produce an edited sequential or partitioned data set or PDSE.
- Manipulate data sets containing double-byte character set data.
- Print sequential data sets or members of partitioned data sets or PDSEs.
- Reblock or change the logical record length of a data set.

Note: If you have the DFSORT product installed, you should be using ICEGENER as an alternative to IEBGENER when making an unedited copy of a data set or member. It may already be installed in your system under the name IEBGENER. It generally gives better performance.

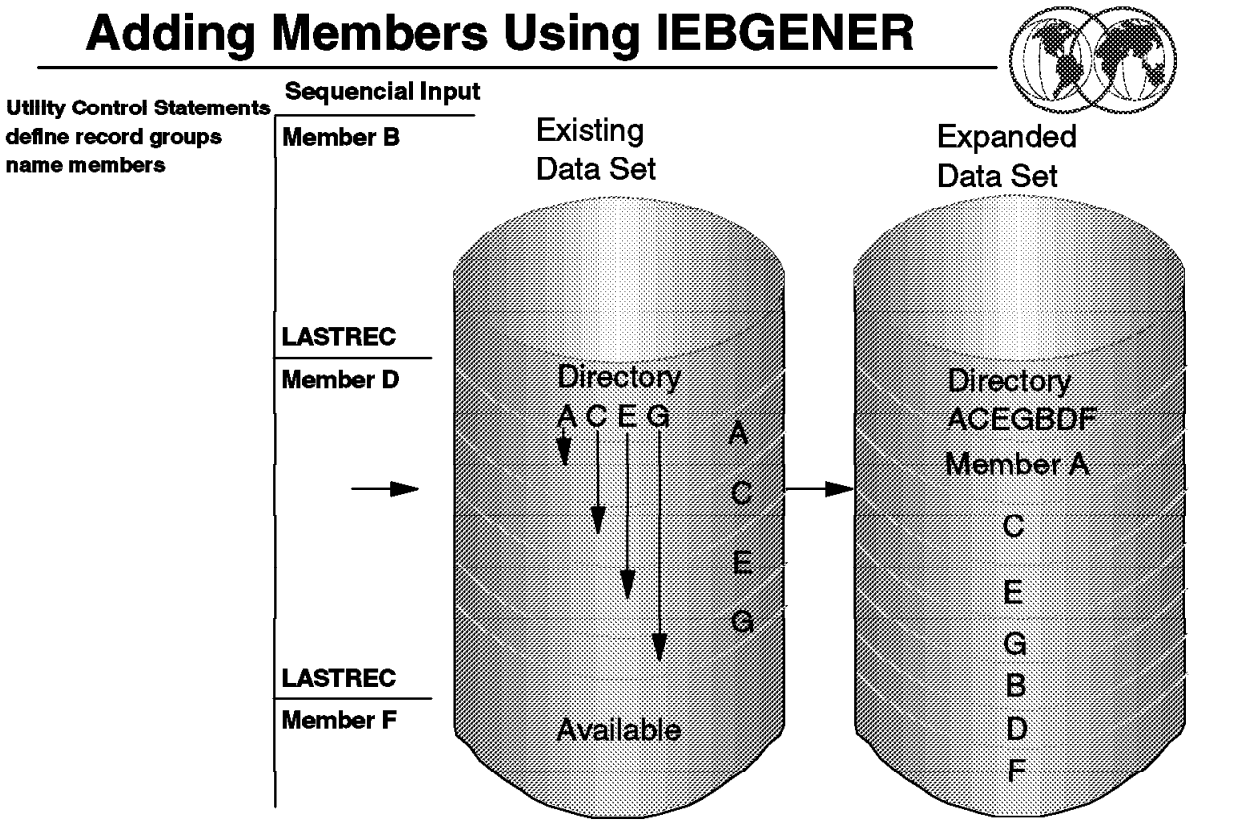


Figure 42. Adding members to a PDS using IEBGENER

1.13.1 Adding members to a PDS using IEBGENER

You can use IEBGENER to add members to a partitioned data set or PDSE. IEBGENER creates the members from sequential input and adds them to the data set. The merge operation—the ordering of the partitioned directory—is automatically performed by the program.

Figure 42 shows how sequential input is converted into members that are merged into an existing partitioned data set or PDSE. The left side of the figure shows the sequential input that is to be merged with the partitioned data set or PDSE shown in the middle of the figure. Utility control statements are used to divide the sequential data set into record groups and to provide a member name for each record group. The right side of the figure shows the expanded partitioned data set or PDSE. Note that members B, D, and F from the sequential data set were placed in available space and that they are sequentially ordered in the partitioned directory

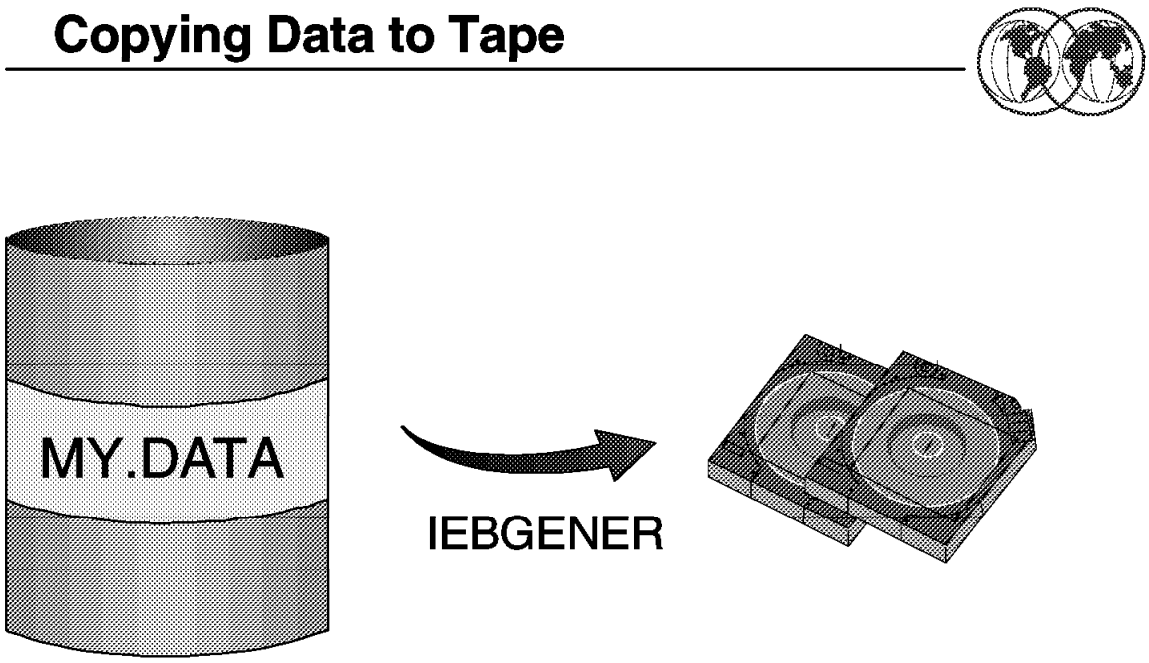


Figure 43. Copying data to tape

1.13.2 Copying data to tape

You can use IEBGENER to copy data to tape. The following example copies the data set MY.DATA to an SL cartridge.

```
//DISKTOTP JOB  ...
//STEP1    EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DSNAME=MY.DATA,DISP=SHR
//SYSUT2   DD  DSNAME=MY.DATA.OUTPUT,UNIT=3490,DISP=(,KEEP),
//             VOLUME=SER=IBM001,LABEL=(1,SL)
//SYSIN    DD  *
```

For further information about IEBGENER, refer to *DFSMS/MVS Utilities*, SC26-4926.

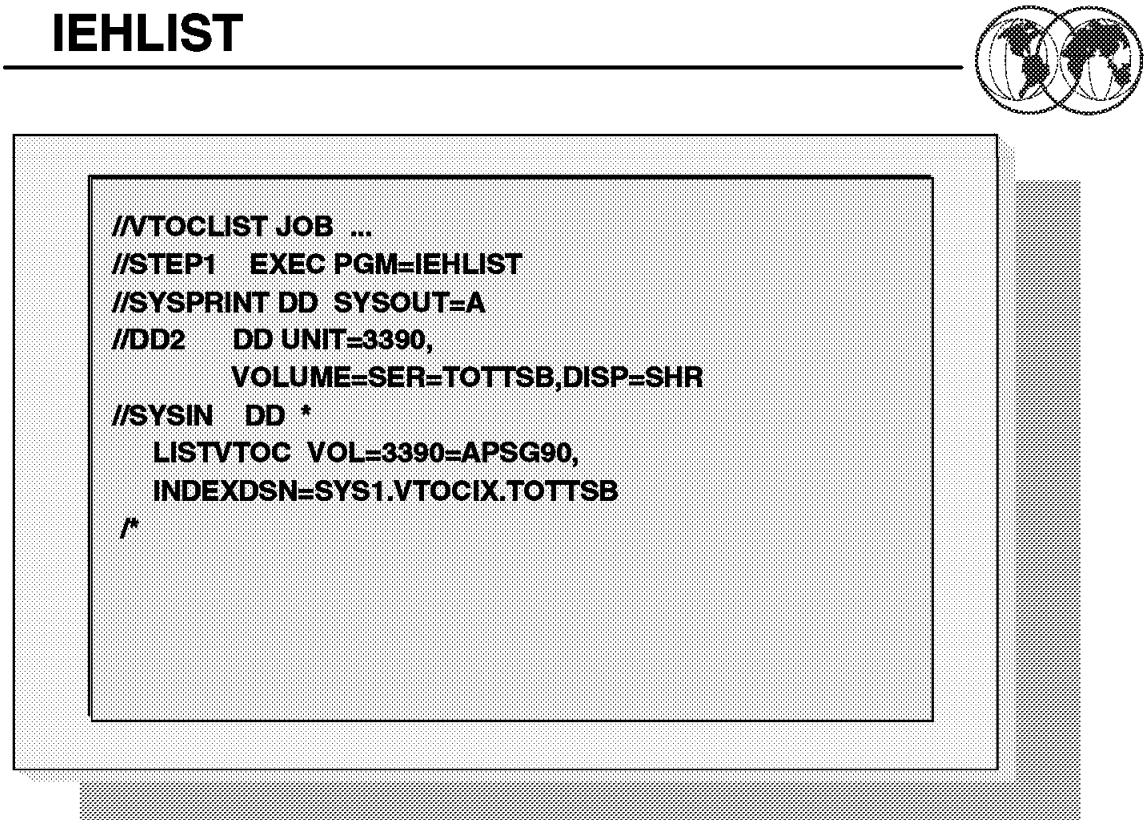


Figure 44. IEHLIST

1.14 IEHLIST

IEHLIST is a system utility used to list entries in a CVOL, entries in the directory of one or more partitioned data sets or PDSEs, or entries in an indexed or non-indexed volume table of contents. Any number of listings can be requested in a single execution of the program.

IEHLIST lists all CVOL (SYSCTLG data set) entries that are part of the structure of a fully qualified data set name.

IEHLIST will not list integrated catalog facility or VSAM catalogs. To list integrated catalog facility or VSAM catalogs, use access method services. For more information, see *DFSMS/MVS Access Method Services for the Integrated Catalog Facility*, SC26-4906.

IEHLIST can list up to 10 partitioned data set or PDSE directories at a time.

The directory of a partitioned data set is composed of variable-length records blocked into 256-byte blocks. Each directory block can contain one or more entries that reflect member or alias names and other attributes of the partitioned members. IEHLIST can list these blocks in edited and unedited format.

The directory of a PDSE, when listed, will have the same format as the directory of a partitioned data set.

IEHLIST LISTVTOC Output

```
  DATE: 1999.118  TIME: 13.35.29
  CONTENTS OF VTOC ON VOL TOTTSB  <THIS VOLUME IS NOT SMS MANAGED>
   THERE IS A  2 LEVEL VTOC INDEX
    DATA SETS ARE LISTED IN ALPHANUMERIC ORDER
-------DATA SET NAME-------   CREATED  DATE.EXP  FILE TYPE
$$WF1                         1998.175   00.000  SEQUENTIAL
$$WK1                         1998.148   00.000  SEQUENTIAL
ALEX.SC61.SPFLOG3.LIST        1998.295   00.000  SEQUENTIAL
ALEXG7.CBC.SPUFI.CBCDBDEF     1998.181   00.000  SEQUENTIAL
ALEXG7.SC42.SPFTEMP2.LIST     1998.187   00.000  SEQUENTIAL
ALEXG7.SC47.SPFLOG1.LIST      1998.286   00.000  SEQUENTIAL
ALEX1.SC61.SPFLOG8.LIST       1998.107   00.000  SEQUENTIAL
ALEX1.SC62.SPFLOG4.LIST       1998.146   00.000  SEQUENTIAL
AMCHENG.ACTEST                1999.082   00.000  SEQUENTIAL
AMCHENG.BRODCA   ST           1999.070   00.000  SEQUENTIAL
AMCHENG.JCL                   1999.109   00.000  SEQUENTIAL
AMCHENG.SC55.ISPF42.ISPPROF   1999.070   00.000  PARTITIONED
AXEL.LOG.MISC                 1999.110   00.000  SEQUENTINAL
BAARES1.SC04.ISPF42.ISPPROF   1998.287   00.000  PARTITIONED
```

Figure 45. LISTVTOC output

1.14.1 LISTVTOC output

Running the job shown in Figure 44 on page 68, you will have a SYSOUT very similar with the one that is shown in Figure 45.

If you include the keyword FORMAT in the LISTVTOC parameter, you will have more detailed information about the DASD and about the data sets, you can also specify the DSNAME that you want to request information about. This information is at DASD volume level, and does not have any interaction with the catalog.

For more detailed information about the utilities explained in this book and other utilities, refer to *DFSMS/MVS Utilities*, SC26-4926.

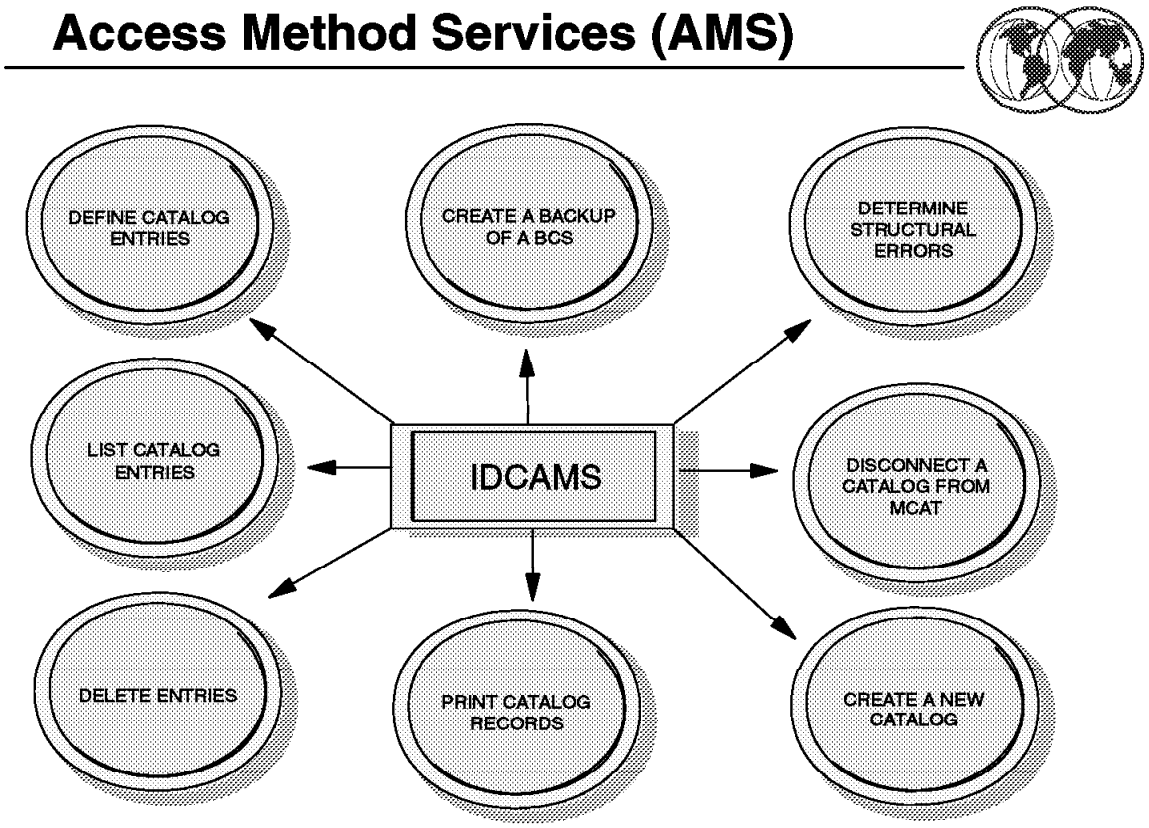


Figure 46. Access method services

1.15 Access method services

Access method services is a utility you can use to establish and maintain catalogs and data sets (VSAM and non-VSAM).

There are two types of access method services commands:

- Functional commands, used to request the actual work (for example, defining a data set or listing a catalog).
- Modal commands that allow the conditional execution of functional commands (it looks like a language). Time sharing option (TSO) users can use functional commands only. For more information about modal commands, refer to *DFSMS/MVS Access Method Services for the Integrated Catalog Facility*, SC26-4906.

The Storage Management Subsystem (SMS) automates many access method services commands and their parameters. The automatic class selection (ACS) routines (established by your storage administrator) and the associated SMS classes eliminate the need to use many access method services command parameters. We will discuss more about the SMS environment later in this chapter.

1.15.1 Invoking access method services

When you want to use an access method services function, enter a command and specify its parameters. Your request is decoded one command at a time; the appropriate functional routines perform all services required by that command.

You can call the access method services program:

- As a job or jobstep
- From a TSO session
- From within your own program

You can run the IDCAMS program (the execution part of access method services) and include the command and its parameters as input to the program. You can also call the IDCAMS program from within another program and pass the command and its parameters to the IDCAMS program.

Time Sharing Option (TSO) users can run access method services functional commands from a TSO session as though they were TSO commands.

For more information, refer to “Invoking Access Method Services from Your Program” in topic D.0, in *DFSMS/MVS Access Method Services for the Integrated Catalog Facility*, SC26-4906.

1.15.1.1 As a job or jobstep

You can use (JCL) statements to call access method services. PGM=IDCAMS identifies the access method services program.

```
//YOURJOB  JOB  YOUR INSTALLATION'S JOB=ACCOUNTING DATA
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *

 access method services commands and their parameters

/*
```

1.15.1.2 From a Time Sharing Option (TSO) session

You can use TSO with VSAM and access method services to:

- Run access method services commands
- Run a program to call access method services

Each time you enter an access method services command as a TSO command, TSO builds the appropriate interface information and calls access method services.

You can enter one command at a time. Access method services processes the command completely before TSO lets you continue processing. Except for ALLOCATE, all the access method services functional commands are supported in a TSO environment.

For more information, refer to *DFSMS/MVS Access Method Service for the Integrated Catalog Facility*, SC26-4906.

1.15.1.3 From within your own program

You can also call the IDCAMS program from within another program and pass the command and its parameters to the IDCAMS program

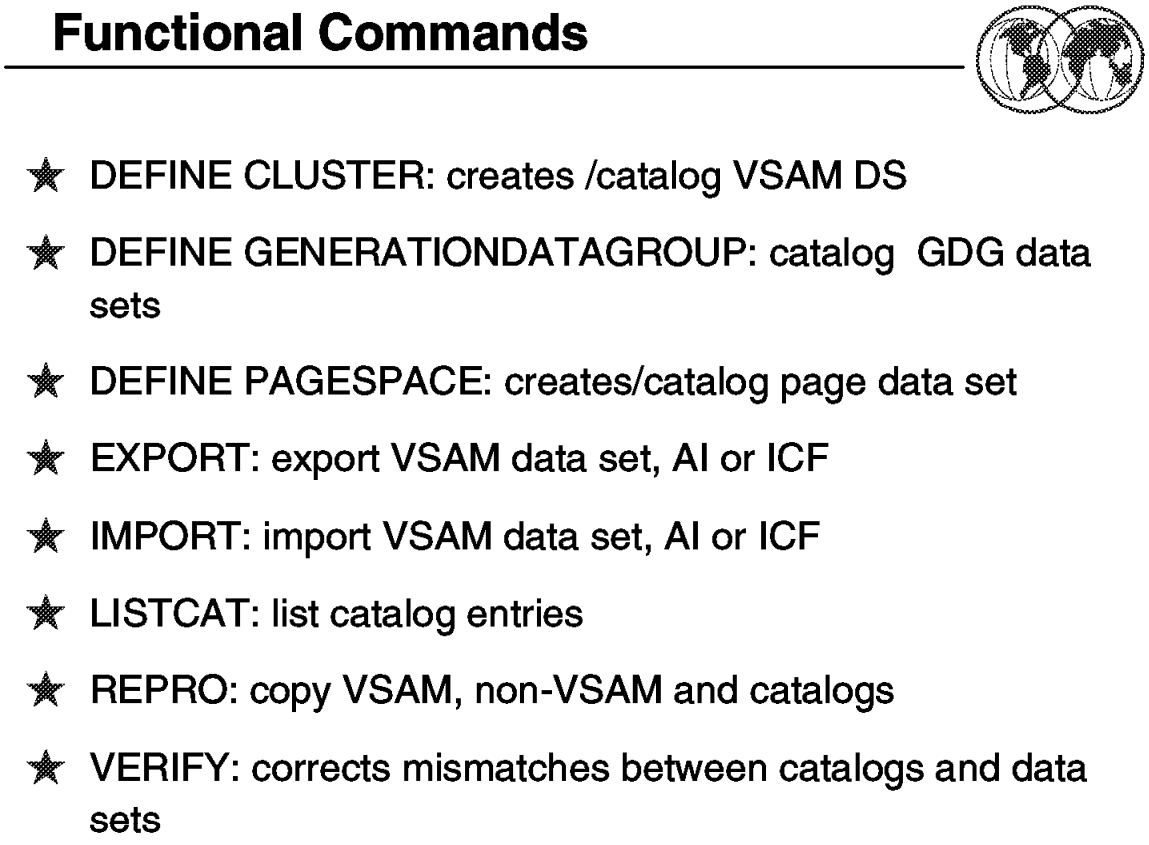


Figure 47. Functional commands

1.15.2 Functional commands

Table 7 describes the utilization of the functional commands.

Table 7 (Page 1 of 2). Functional commands

Command	Function
ALLOCATE	Creates VSAM and non-VSAM data sets.
ALTER	Alters attributes of data sets, catalogs, tape library entries, and tape volume entries that have already been defined.
BLDINDEX	Builds alternate indexes (AI) for existing data sets.
CNVTCAT	Converts VSAM catalog and CVOL entries to integrated catalog facility catalog entries.
CREATE	Creates tape library entries and tape volume entries.
DCOLLECT	Collects data set, volume usage, and migration utility information.
DEFINE ALIAS	Defines an alternate name for a non-VSAM data set or a user catalog. In AMS jargon defines means create and when applied to VSAM data sets implies catalog (after the creation) as well.
DEFINE ALTERNATEINDEX	Defines an alternate index in the catalog.
DEFINE CLUSTER	Defines (creates and catalogs) a cluster for an entry-sequenced, key-sequenced, linear, or relative record data set.

Table 7 (Page 2 of 2). Functional commands	
Command	**Function**
DEFINE GENERATIONDATAGROUP	Defines a catalog entry for a generation data group.
DEFINE NONVSAM	Defines a catalog entry for a non-VSAM data set
DEFINE PAGESPACE	Defines an entry for a page space data set.
DEFINE PATH	Defines a path directly over a base cluster or over an alternate index and its related base cluster.
USERCATALOG / MASTERCATALOG	Defines a catalog.
DELETE	Deletes catalogs, VSAM data sets, and non-VSAM data sets.
DIAGNOSE	Scans an integrated catalog facility basic catalog structure (BCS) or a VSAM volume data set (VVDS) to validate the data structures and detect structure errors.
EXAMINE	Analyzes and reports the structural consistency of either an index or data component of a key-sequence data set cluster.
EXPORT	Exports VSAM data sets (including alternate index) and integrated catalog facility backups. Export means the creation of a non-executable copy of the VSAM entity. This copy is suitable to be recorded in tape to be exported to another data center or a safe. To process data that was exported, an import is a must.
EXPORT DISCONNECT	Export disconnects a user catalog.
IMPORT	Imports VSAM data sets and integrated catalog facility catalogs.
IMPORT CONNECT	Connects a user catalog or a volume catalog.
LISTCAT	Lists catalog entries
PRINT	Used to print VSAM data sets, non-VSAM data sets, and catalogs.
REPRO	Performs the following functions: • Copies VSAM and non-VSAM data sets, user catalogs, master catalogs, and volume catalogs. • Splits integrated catalog facility catalog entries between two catalogs. • Merges integrated catalog facility catalog entries into another integrated catalog facility user or master catalog. • Merges tape library catalog entries from one volume catalog into another volume catalog.
SHCDS	Lists SMSVSAM recovery related to online applications and spheres accessed in RLS mode.
VERIFY	Causes a catalog to correctly reflect the end of a data set after an error occurred while closing a VSAM data set. The error might have caused the catalog to be incorrect.

For a complete description of all AMS commands, refer to *DFSMS/MVS Access Method Service for the Integrated Catalog Facility*, SC26-4906.

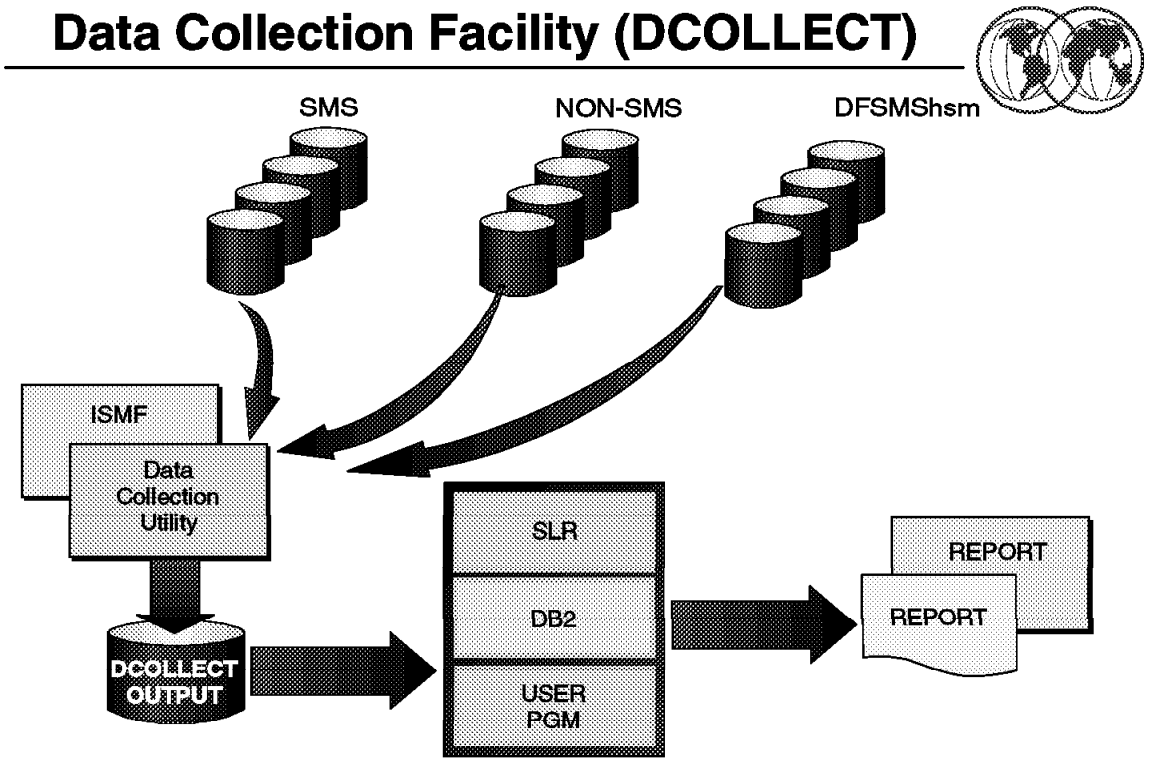


Figure 48. Data Collection Facility (DCOLLECT)

1.16 Data Collection Facility (DCOLLECT)

DCOLLECT is an IDCAMS command. ISMF provides the option to build the JCL necessary to execute DCOLLECT.

An installation may collect information related to:

- Active data set storage
- VSAM association name
- Volume usage
- DFSMShsm backup and migration storage
- DFSMShsm DASD and tape capacity planning

Data is gathered from the VTOC, VVDS, and DFSMShsm control data set for both managed and non-managed storage.

The output of DCOLLECT is a sequential data set. To generate reports from the collected data relating to space management, capacity planning, and cost accounting, consider:

- SLR (Service Level Reporter) V3.2 with APAR PL54270 provides support for DCOLLECT data by including a starter set of log, summary, and parameter tables and views. It also includes a report dialog with a variety of predefined reports.

- DB2 can also be used to hold DCOLLECT data. Data resides in a relational database structure, and can be presented to users in a table format.
- User written applications can manipulate DCOLLECT's sequential output data set to generate reports.

For more information, see the following publications:

- *DFSMS/MVS Access Method Service for the Integrated Catalog Facility*, SC26-4906
- *Service Level Reporter User's Guide: Reporting*, SH19-6530

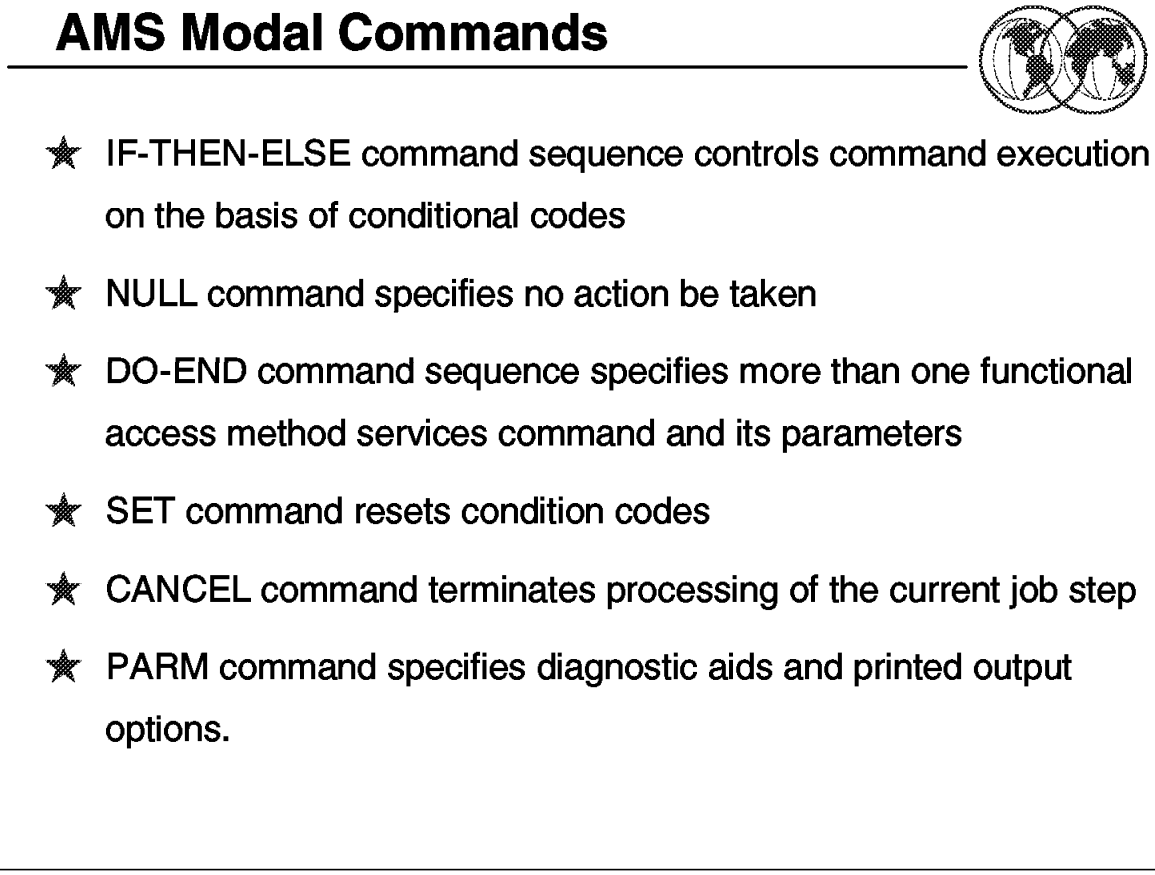


Figure 49. AMS modal commands

1.16.1 AMS modal commands

There are two types of access method services commands: functional commands, used to request the actual work (for example, defining a data set or listing a catalog), and modal commands that allow the conditional execution of functional commands. Time Sharing Option (TSO) users can use functional commands only.

This visual shows a brief description of the AMS modal commands. These commands cannot be used when access method services is run in TSO. See *DFSMS/MVS Access Method Services for the Integrated Catalog Facility*, SC26-4906, for a complete description of the AMS modal commands.

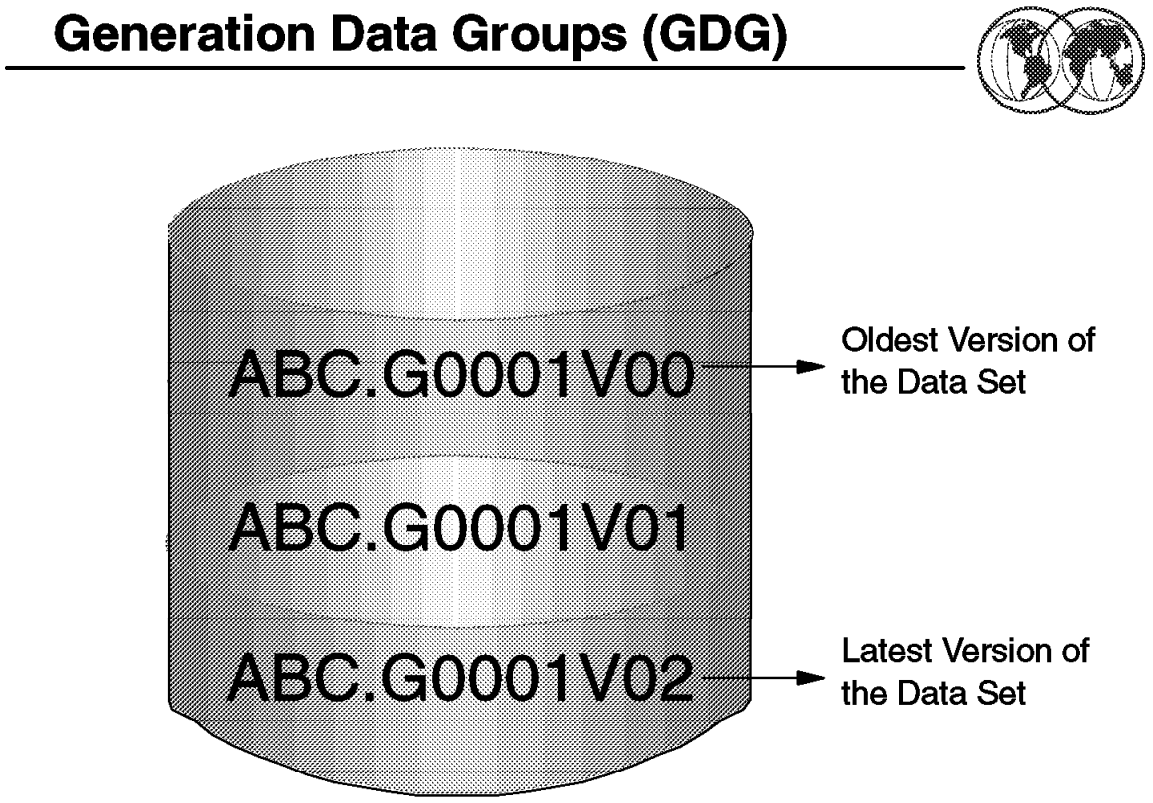


Figure 50. Generation data groups

1.17 Generation data groups (GDG)

You can catalog successive updates or generations of related data sets. They are called generation data groups (GDG). Each data set within a GDG is called a generation data set or generation. Within a GDG, the generations can have like or unlike DCB attributes and data set organizations. If the attributes and organizations of all generations in a group are identical, the generations can be retrieved together as a single data set.

Generation data sets can be sequential, direct, or indexed sequential (an old and less used data set organization, replaced by VSAM KSDS). They cannot be partitioned, HFS, or VSAM. The same GDG may contain SMS and non-SMS data sets.

There are advantages to grouping related data sets. For example, the catalog management routines can refer to the information in a special index called a generation index in the catalog. Thus:

- All of the data sets in the group can be referred to by a common name.
- The operating system is able to keep the generations in chronological order.
- Outdated or obsolete generations can be automatically deleted by the operating system.

Another advantage is the ability to reference to a new generation using the same JCL.

Generation data sets have sequentially ordered absolute and relative names that represent their age. The catalog management routines use the absolute generation name. Older data sets have smaller absolute numbers. The relative name is a signed integer used to refer to the latest (0), the next to the

latest (-1), and so forth, generation. For example, a data set name LAB.PAYROLL(0) refers to the most recent data set of the group; LAB.PAYROLL(-1) refers to the second most recent data set; and so forth. The relative number can also be used to catalog a new generation (+1).

If you create a generation data set with a relative generation number of (+1), the system recognizes any subsequent reference to (+1) throughout the job as having the same absolute generation number.

A GDG base is allocated in an integrated catalog facility or VSAM catalog before the generation data sets are cataloged. Each GDG is represented by a GDG base entry. Use the AMS DEFINE command to allocate the GDG base.

The model DSCB must exist on the GDG catalog volume.

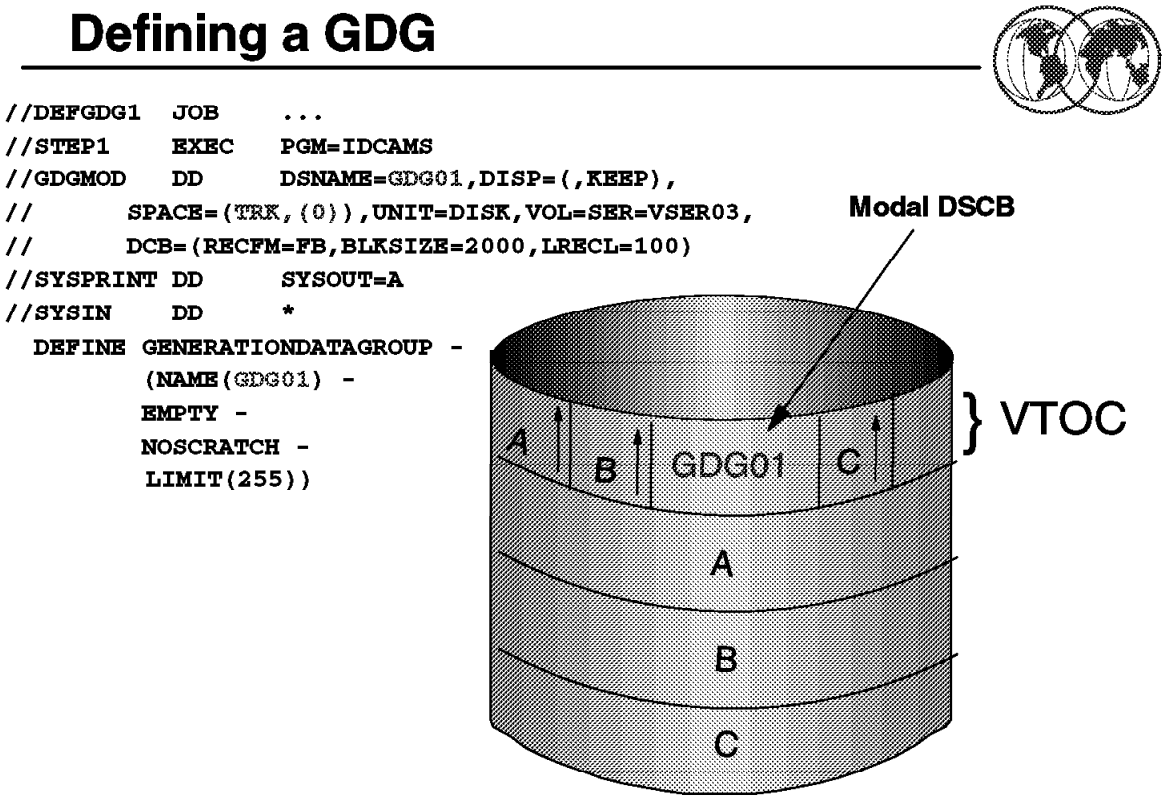


Figure 51. Defining a GDG

1.17.1 Defining a generation data group

The DEFINE GENERATIONDATAGROUP command creates a catalog entry for a generation data group (GDG).

Following is an example of a DEFINE GDG.

```
//DEFGDG1  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//GDGMOD   DD     DSNAME=GDG01,DISP=(,KEEP),
//         SPACE=(TRK,(0)),UNIT=DISK,VOL=SER=VSER03,
//         DCB=(RECFM=FB,BLKSIZE=2000,LRECL=100)
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
    DEFINE GENERATIONDATAGROUP -
           (NAME(GDG01) -
           NOEMPTY -
           NOSCRATCH -
           LIMIT(255) )
/*
```

The DEFINE GENERATIONDATAGROUP command defines a GDG base catalog entry GDG01. Its parameters are:

- NAME specifies the name of the GDG, GDG01. Each GDS in the group will have the name *GDG01.GxxxxVyy*, where *xxxx* is the generation number and *yy* is the version number.

- NOEMPTY specifies that only the oldest generation data set is to be uncataloged when the maximum is reached (recommended).
- EMPTY specifies that all data sets in the group are to be uncataloged by VSAM when the group reaches the maximum number of data sets (as specified by the LIMIT parameter) and one more GDS is added to the group.
- NOSCRATCH specifies that when a data set is uncataloged, its DSCB is not to be removed from its volume′s VTOC. Therefore, even if a data set is uncataloged, its records can be accessed when it is allocated to a job step with the appropriate JCL DD statement.
- LIMIT specifies that the maximum number of GDG data sets in the group is 255. The LIMIT parameter is required.

Next, a generation data set is defined within the GDG by using JCL statements.

```
//DEFGDG2  JOB    ...
//STEP1    EXEC   PGM=IEFBR14
//GDGDD1   DD     DSNAME=GDG01(+1),DISP=(NEW,CATLG),
//         SPACE=(TRK,(10,5)),VOL=SER=VSER03,
//         UNIT=DISK
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
/*
```

The job DEFGDG2, allocates space and catalogs a GDG data set in the newly-defined GDG. The job control statement GDGDD1 DD specifies the GDG data set in the GDG.

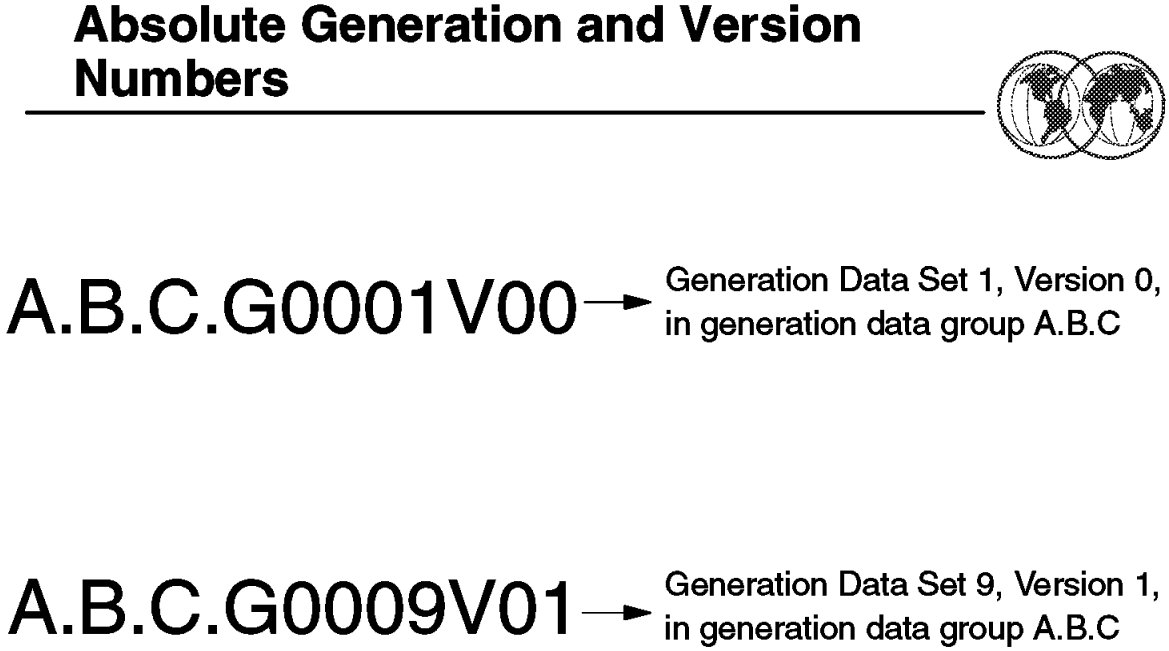


Figure 52. Absolute generation and version numbers

1.17.2 Absolute generation and version numbers

An absolute generation and version number is used to identify a specific generation of a generation data group. A same generation data set may have different versions, which are maintained by your installation. The version number allows you to perform normal data set operations without disrupting the management of the generation data group. For example, if you want to update the second generation in a three-generation group, replace generation 2, version 0, with generation 2, version 1. Only one version is kept for each generation.

The generation and version number are in the form *GxxxxVyy*, where *xxxx* is an unsigned four-digit decimal generation number (0001 through 9999) and *yy* is an unsigned two-digit decimal version number (00 through 99). For example:

- A.B.C.G0001V00 is generation data set 1, version 0, in generation data group A.B.C.
- A.B.C.G0009V01 is generation data set 9, version 1, in generation data group A.B.C.

The number of generations and versions is limited by the number of digits in the absolute generation name; that is, there can be 9,999 generations. Each generation can have 100 versions.

The system automatically maintains the generation number. The number of generations kept depends on the size of the generation index. For example, if the size of the generation index allows ten entries (parameter LIMIT in AMS DEFINE), the ten latest generations can be maintained in the generation data group (parameter NOEMPTY in AMS DEFINE).

You can catalog a generation using either absolute or relative numbers. When a generation is cataloged, a generation and version number is placed as a low-level entry in the generation data group. To catalog a version number other than V00, you must use an absolute generation and version number.

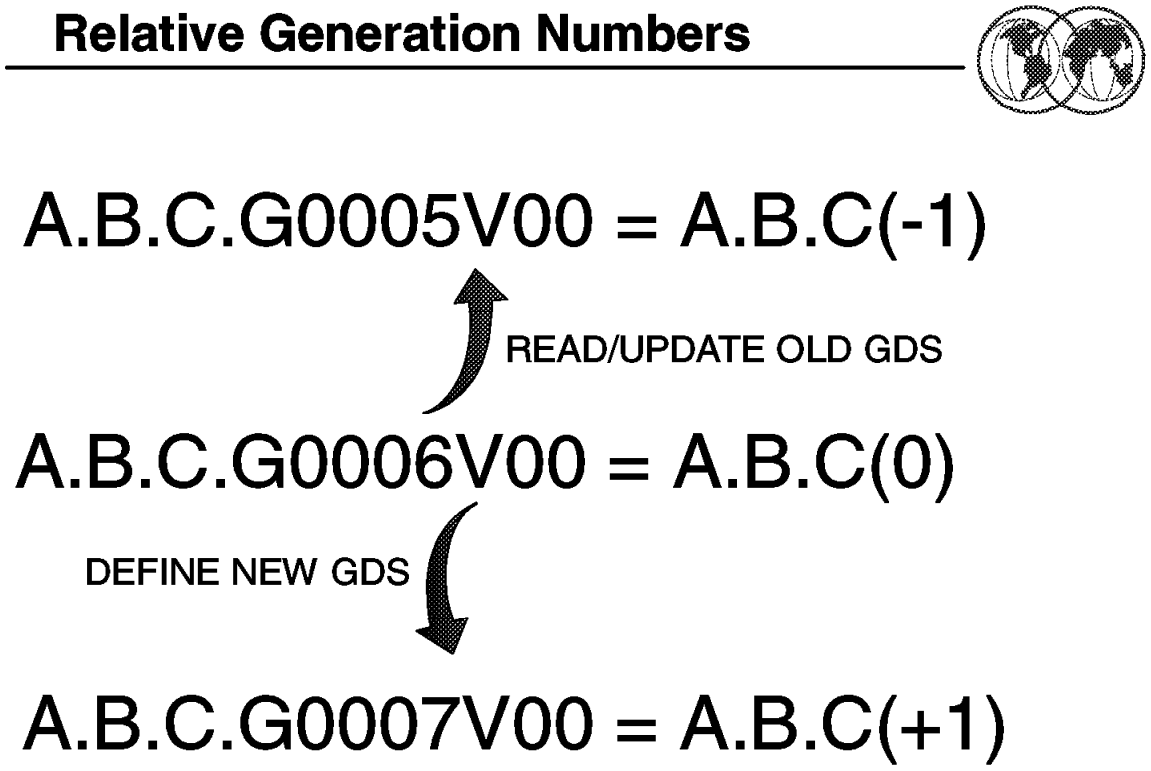


Figure 53. Relative generation numbers

1.17.3 Relative generation number

As an alternative to using absolute generation and version numbers when cataloging or referring to a generation, you can use a relative generation number. To specify a relative number, use the generation data group name followed by a negative integer, a positive integer, or a zero (0), enclosed in parentheses. For example, A.B.C(-1). A.B.C(+1), or A.B.C(0).

The value of the specified integer tells the operating system what generation number to assign to a *new generation data set*, or it tells the system the location of an entry representing a previously cataloged *old generation data set*.

When you use a relative generation number to catalog a generation, the operating system assigns an absolute generation number and a version number of V00 to represent that generation. The absolute generation number assigned depends on the number last assigned and the value of the relative generation number that you are now specifying. For example if, in a previous job generation, A.B.C.G0006V00 was the last generation cataloged, and you specify A.B.C(+1), the generation now cataloged is assigned the number G0007V00.

Though any positive relative generation number can be used, a number greater than 1 can cause absolute generation numbers to be skipped for a new generation data set. For example, if you have a single step job, and the generation being cataloged is a +2, one generation number is skipped. However, in a multiple step job, one step might have a +1 and a second step a +2, in which case no numbers are skipped.

The mapping between relative and absolute numbers are kept until the end of the job

1.17.4 Rolled in and rolled off

When a generation data group contains its maximum number of active generation data sets, defined in the LIMIT parameter, and a new generation data set is rolled in at end-of-job step, the oldest generation data set is rolled off and is no longer active. If a generation data group is defined using DEFINE GENERATIONDATAGROUP EMPTY, and is at its limit, then, when a new generation data set is rolled in, all the currently active generation data sets are rolled off.

The parameters you specify on the DEFINE GENERATIONDATAGROUP command determines what happens to rolled off generation data sets. For example, if you specify the SCRATCH parameter, the generation data set is scratched when it is rolled off. If you specify the NOSCRATCH parameter, the rolled off generation data set is re-cataloged as rolled off and is disassociated with its generation data group.

Generation data sets can be in a deferred roll-in state if the job never reached end-of-step or if they were allocated as DISP=(NEW,KEEP) and the data set is not system-managed. Generation data sets in a deferred roll-in state can be referred to by their absolute generation numbers. You can use the access method service command ALTER ROLLIN to roll in these generation data sets.

For further information about Generation Data Groups, refer to *DFSMS/MVS Using Data Sets*, SC26-4922.

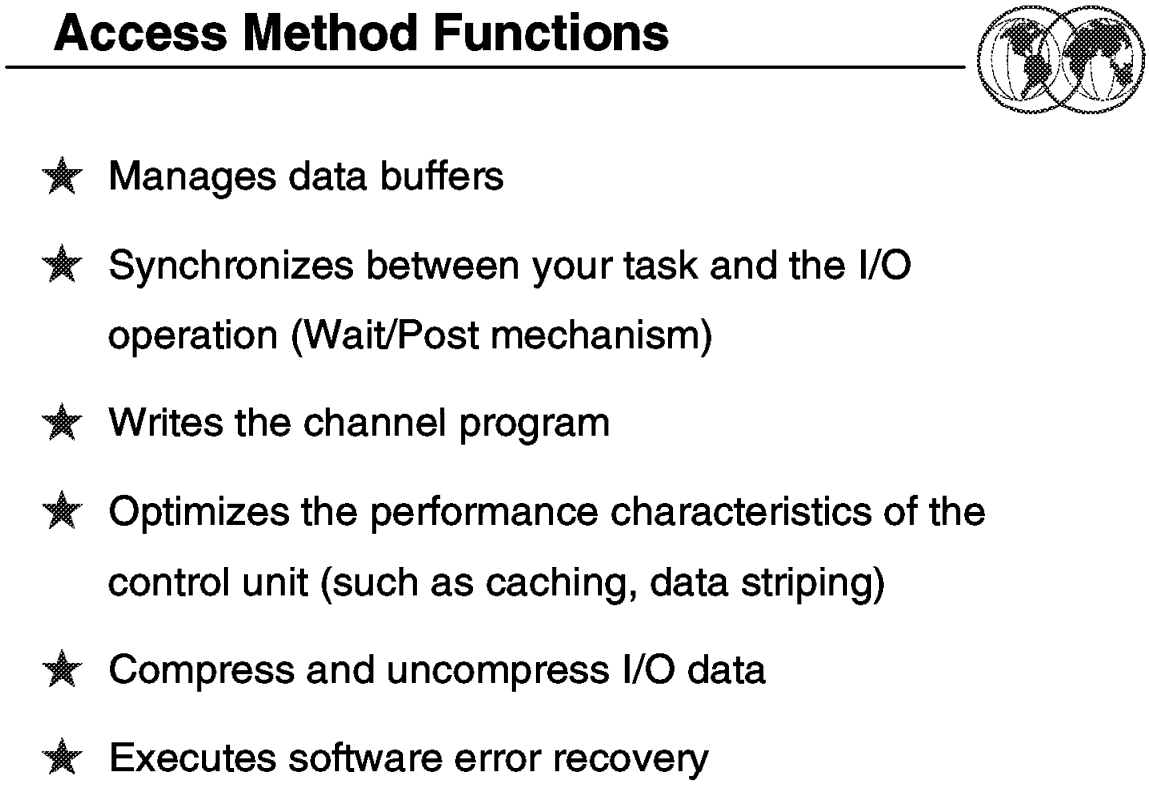


Figure 54. Access method functions

1.18 Access method functions

An access method is a friendly interface between programs and their data. It is in charge of interfacing with Input Output Supervisor (IOS), and the MVS code which starts the I/O operation. An access method:

- Makes transparent to you the physical organization of data, by:
  - Managing data buffers
  - Synchronizing your task and the I/O operation (Wait/Post mechanism)
  - Writing the channel program
  - Optimizing the performance characteristics of the control unit (such as caching and data striping)
  - Compressing and decompressing I/O data
- Executes software error recovery

An access method defines the technique by which the data is stored and retrieved. DFSMS/MVS access methods have their own data set structures for organizing data, macros to define and process data sets, and utility programs to process data sets.

Access methods are identified primarily by the data set organization to which they apply. For example, you can use the basic sequential access method (BSAM) with sequential data sets. However, there are times when an access method identified with one organization can be used to process a data set

organized in a different manner. For example, a sequential data set (not extended format data set) created using BSAM can be processed by the basic direct access method (BDAM), and vice versa.

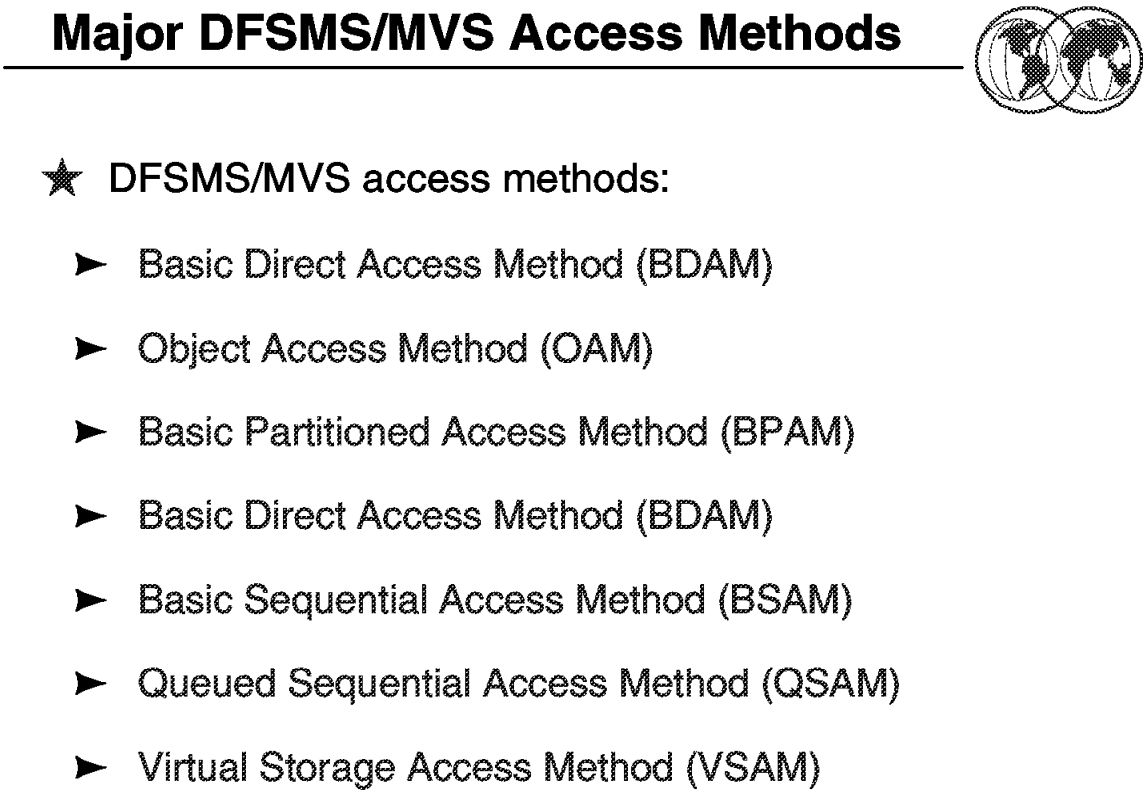


Figure 55. Major DFSMS/MVS access methods

1.18.1 Major DFSMS/MVS access methods

Basic Direct Access Method (BDAM) arranges records in any sequence your program indicates, and retrieves records by actual or relative address. If you do not know the exact location of a record, you can specify a point in the data set where a search for the record is to begin. Data sets organized this way are called direct data sets.

IBM does not recommended using BDAM because it tends to require using device-dependent code. In addition, using keys is much less efficient than in virtual sequential access method (VSAM). BDAM is supported by DFSMS/MVS only to enable compatibility with other IBM operating systems. Appendix C, “Processing Direct Data Sets” in *DFSMS/MVS Using Data Sets*, SC26-4922.

Object Access Method (OAM) processes very large named byte streams (objects) that have no record boundary or other internal orientation. These objects can be recorded in a DB2 data base or on an optical storage volume. For information on OAM, see *DFSMS/MVS Object Access Method Application Programmer's Reference*, SC26-4917, and *DFSMS/MVS Object Access Method Planning, Installation, and Storage Administration Guide for Object Support*, SC26-4918.

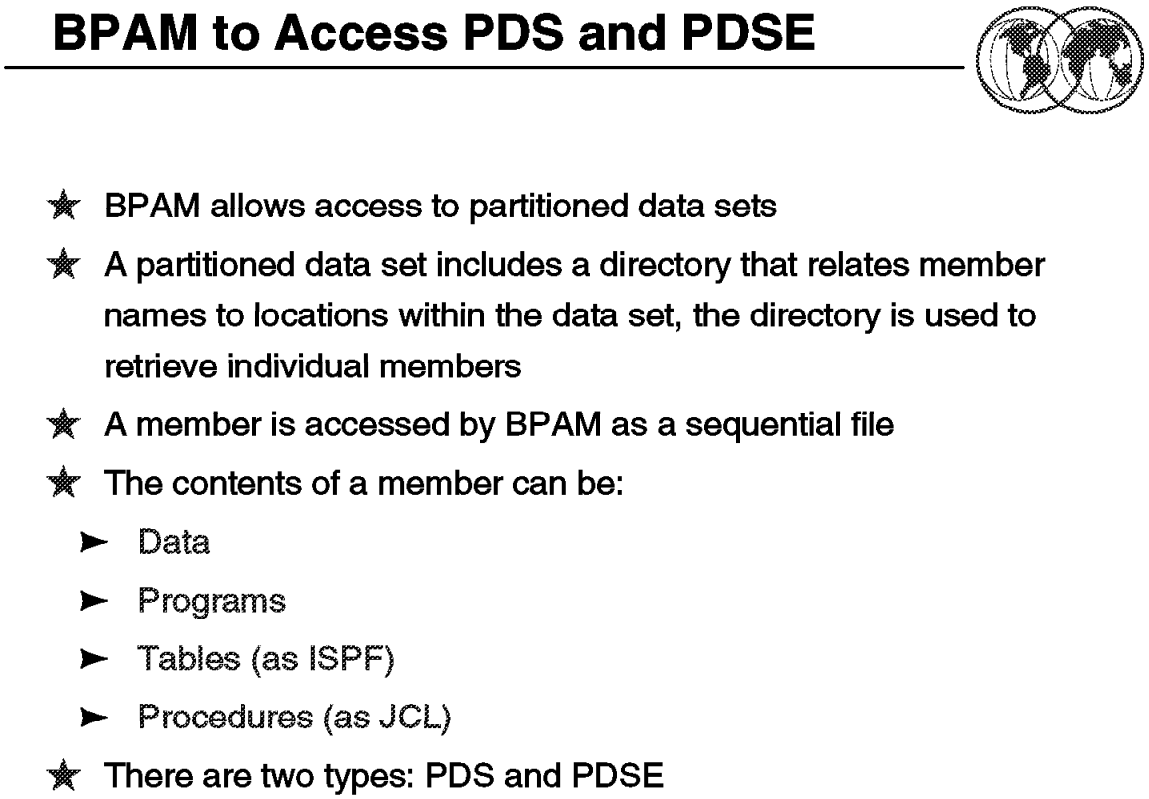


Figure 56. BPAM and PDSE

1.18.2 BPAM to access PDS and PDSE

Basic Partitioned Access Method (BPAM) arranges records as members of a partitioned data set (PDS) or a partitioned data set extended (PDSE) on DASD. You can view each member like a sequential data set. A partitioned data set or PDSE includes a directory that relates member names to locations within the data set. The directory is used to retrieve individual members, and for program libraries (load modules and program objects) contains program attributes required to load and re-bind the member.

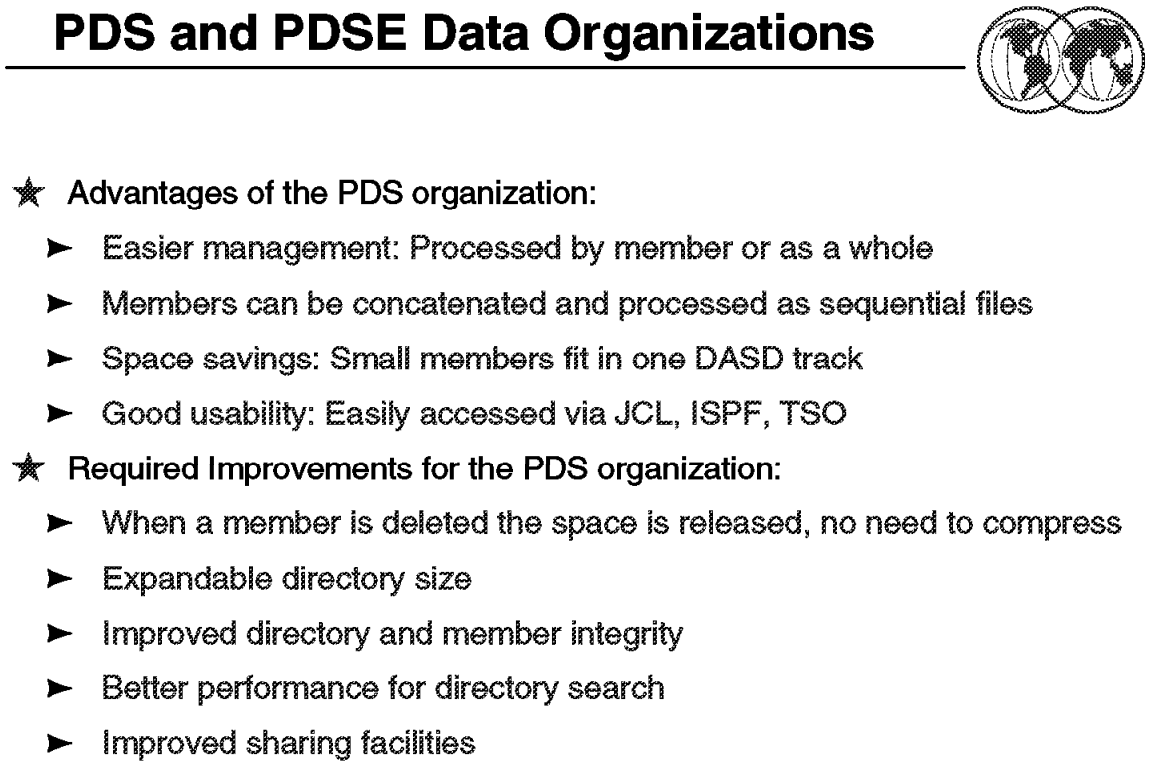


Figure 57. PDS and PDSE data organization

1.18.3 PDS and PDSE data organizations

Partitioned data set (PDS) is an old MVS data organization, which has good features such as:

- *Easier management:* Grouping of related data sets under a single name makes MVS data management easier. Files stored as members of a PDS can be processed either individually or all the members can be processed as a unit.
- *Space savings:* Small members fit in just one DASD track.
- *Good usability:* Members of a PDS can be used as sequential data sets, and they can be concatenated to sequential data sets. They are also easy to create with JCL, or ISPF; they are easy to manipulate with ISPF utilities or TSO commands.

However, there are a few requirements for improvement regarding the PDS organization:

- There is no mechanism to reuse the area which contained a deleted or re-written member. This unused space must be reclaimed by the use of the IEBCOPY utility function called compression.
- Directory size is not expandable, causing an overflow exposure. The area for members may grow using secondary allocations. This is not true for the directory.
- A PDS has no mechanism to stop the directory from being overwritten if a program mistakenly opens it for sequential output. If it happens, the directory is destroyed, and all the members are lost. Also, PDS DCB attributes can be easily changed by mistake. If you add a member whose DCB characteristics differ from those of the other members, you will change the DCB attributes of the entire PDS, and all the old members will become unusable.

- Better directory search time. Entries in the directory are physically ordered by the collating sequence of the names in the members they are pointing to. Any inclusion may cause the full rearrange of the entries. There is also no index to the directory entries. The search is sequential using a CKD format. If the directory is big, the I/O operation takes more time.
- Improved sharing facilities. To update a member of a PDS, you need exclusive access to the entire data set.

All these improvements require almost total compatibility at program and user level with the old PDS.

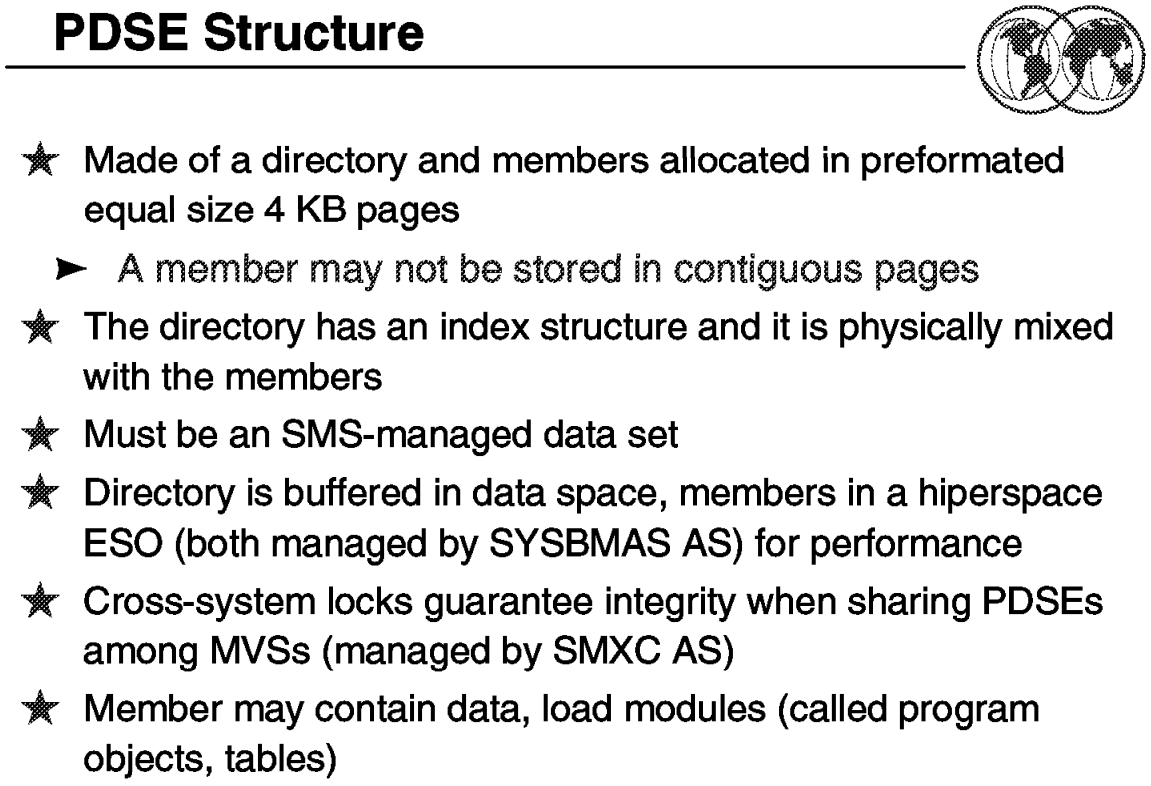


Figure 58. PDSE structure

1.18.4 PDSE structure

The advantages of PDSE when compared with PDS are:

- Space is reclaimed without a compress. PDSE automatically reuses space, without needing an IEBCOPY compress. A list of available space is kept in the directory. When a PDSE member is updated or replaced, it is written in the first available space. This is either at the end of the data set or in a space in the middle of the data set marked for reuse. This space need not be contiguous. The objective of the space reuse algorithm is not to extend the data set unnecessarily.
- The directory can grow dynamically as the data set expands. Logically, a PDSE directory looks the same as a PDS directory. It consists of a series of directory records in a block. Physically, it is a set of pages at the front of the data set, plus additional pages interleaved with member pages. Five directory pages are initially created at the same time as the data set. New directory pages are added, interleaved with the member pages, as new directory entries are required. A PDSE always occupies at least five pages of storage. The directory is like a KSDS index structure, making a search much faster. It cannot be overwritten by being opened for sequential output.
- If you try to add a member with DCB characteristics that differ from the rest of the members, you will get an error.
- You can open a PDSE member for output or update, without locking the entire data set. The sharing control is at member level, not the data set level.

There is a restriction about PDSEs, that is, you cannot use a PDSE for certain system data sets which are opened at the IPL/NIP time frame.

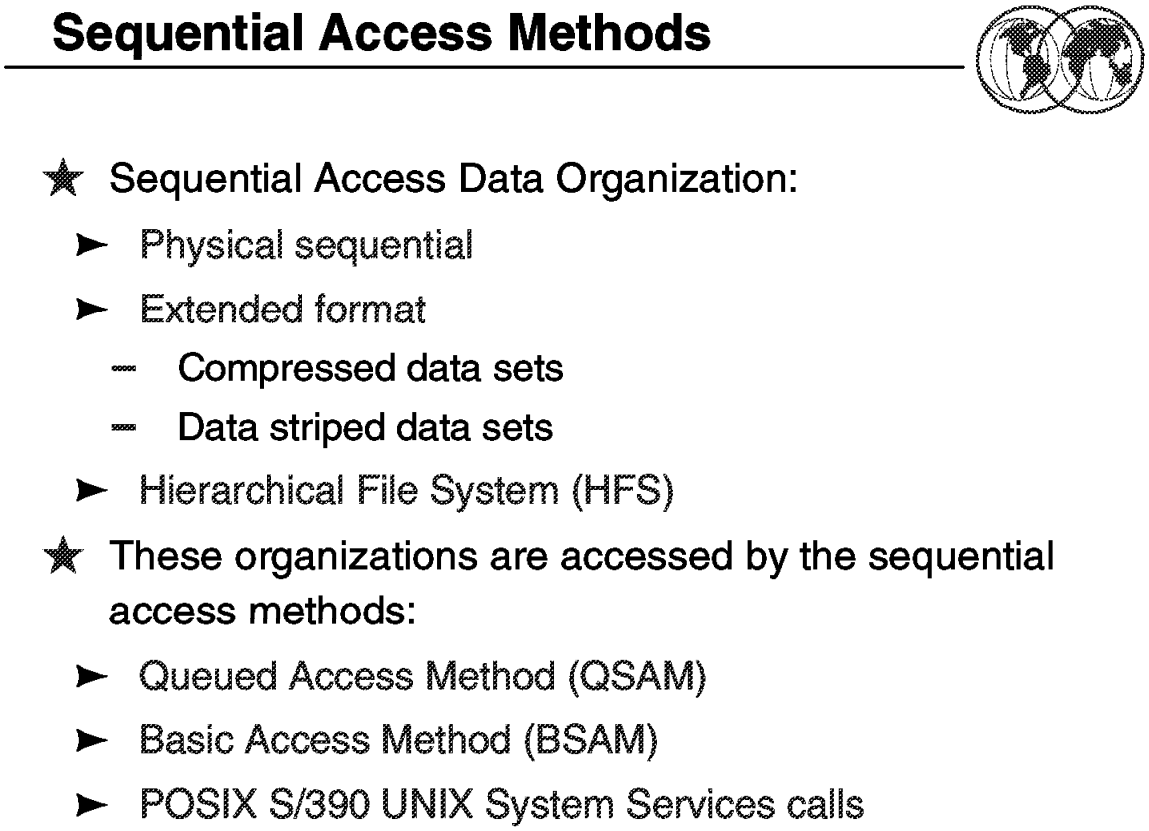


Figure 59. Sequential access methods

1.18.5 Sequential access methods

There are two sequential access methods, Basic Sequential Access Method (BSAM) and Queued Sequential Access Method (QSAM). Both access data organized in a physical sequenced manner. In this manner the physical records (containing logical records) are stored sequentially in the order in which they are entered.

One special sort of this organization is called *Extended Format Data Set*. Extended format data sets have a different internal storage format from a sequential data set that is not extended (fixed block with a 32-bytes suffix). This storage format gives extended format data sets additional usability and availability characteristics:

- Can be allocated in the compressed format (can be referred to as a compressed format data set). A compressed format data set is a type of extended format data set that has an internal storage format that allows for data compression.
- Allows data stripping, that is a multivolume sequential file where data may be accessed in parallel.
- Is able to recover from an padding error situation.

Extended format data sets must be SMS-managed and must reside on DASD. You cannot use an extended format data set for certain system data sets.

Another type of this organization is called *Hierarchical File System*. HFS files are POSIX-conforming files which reside in an HFS data set. They are byte-oriented rather than record-oriented, as are MVS files. They are identified and accessed by specifying the *path* leading to them. Programs can access

the information in HFS files through OS/390 UNIX system calls, such as open(pathname), read(file descriptor), and write(file descriptor). Programs can also access the information in HFS files through the MVS BSAM, QSAM, and VSAM access methods. When using BSAM or QSAM, an HFS file is simulated as a multi-volume sequential data set. When using VSAM, an HFS file is simulated as an ESDS. HFS data sets are:

- Supported by standard DADSM create, rename, and scratch
- Supported by DFSMShsm for dump/restore and migrate/recall if DFSMSdss is used as the data mover
- Not supported by IEBCOPY or the DFSMSdss COPY function

The difference between QSAM and BSAM are:

- QSAM deblocks logical records and does look ahead reads (anticipates reads). In BSAM these tasks are done by the calling program.
- QSAM synchronizes the task with I/O operation (places the task in wait along the I/O operation). In BSAM this tasks is done by the calling program (macro CHECK).

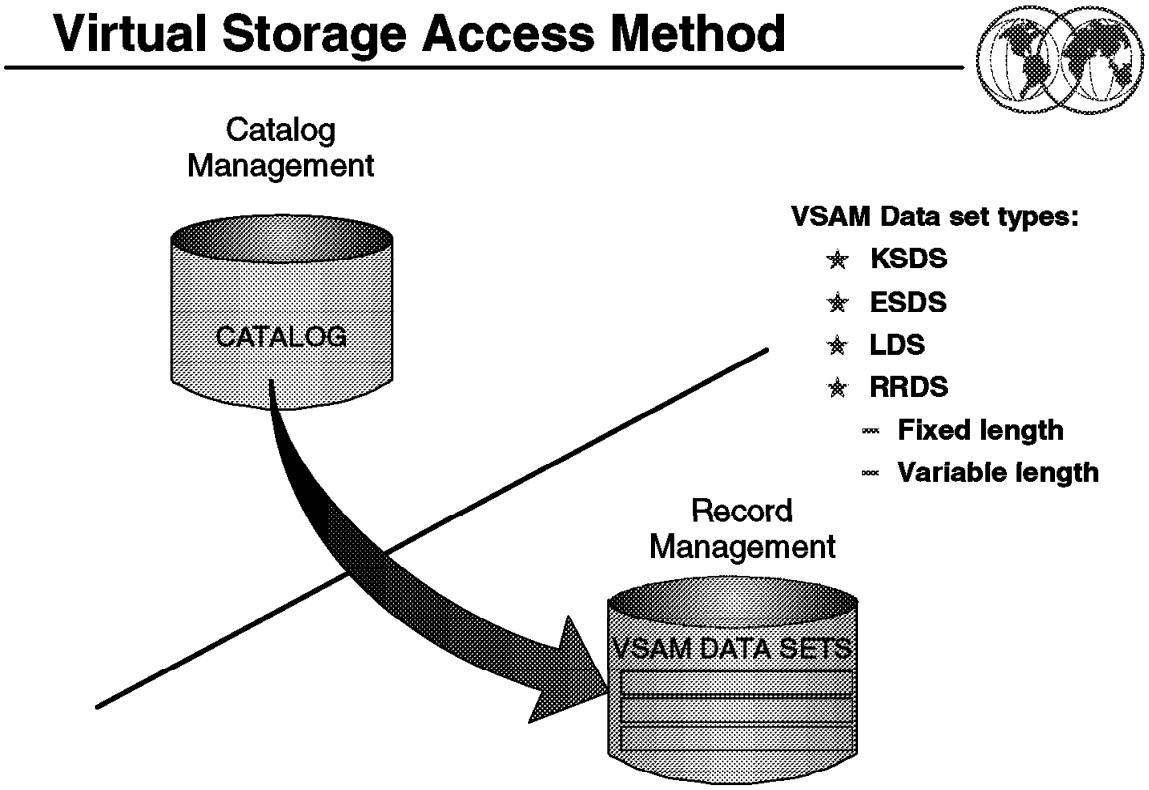


Figure 60. Virtual Storage Access Method

1.19 Virtual Storage Access Method (VSAM)

VSAM is an access method service used to organize data and maintain information about the data in a catalog.

There are two major parts of VSAM:

- Catalog Management: The Catalog contain information about the data sets
- Record management: VSAM can be used to organize records into four types of data sets:
  - Key-sequenced (KSDS)
  - Entry-sequenced (ESDS)
  - Linear (LDS)
  - Relative record with fixed or variable length (RRDS)

  The primary difference among these types of data sets is the way in which their records are stored and accessed.

VSAM arranges records by an index key, by relative byte address, or by relative record number. Data organized by VSAM is cataloged for easy retrieval and is stored in one of four types of data sets.

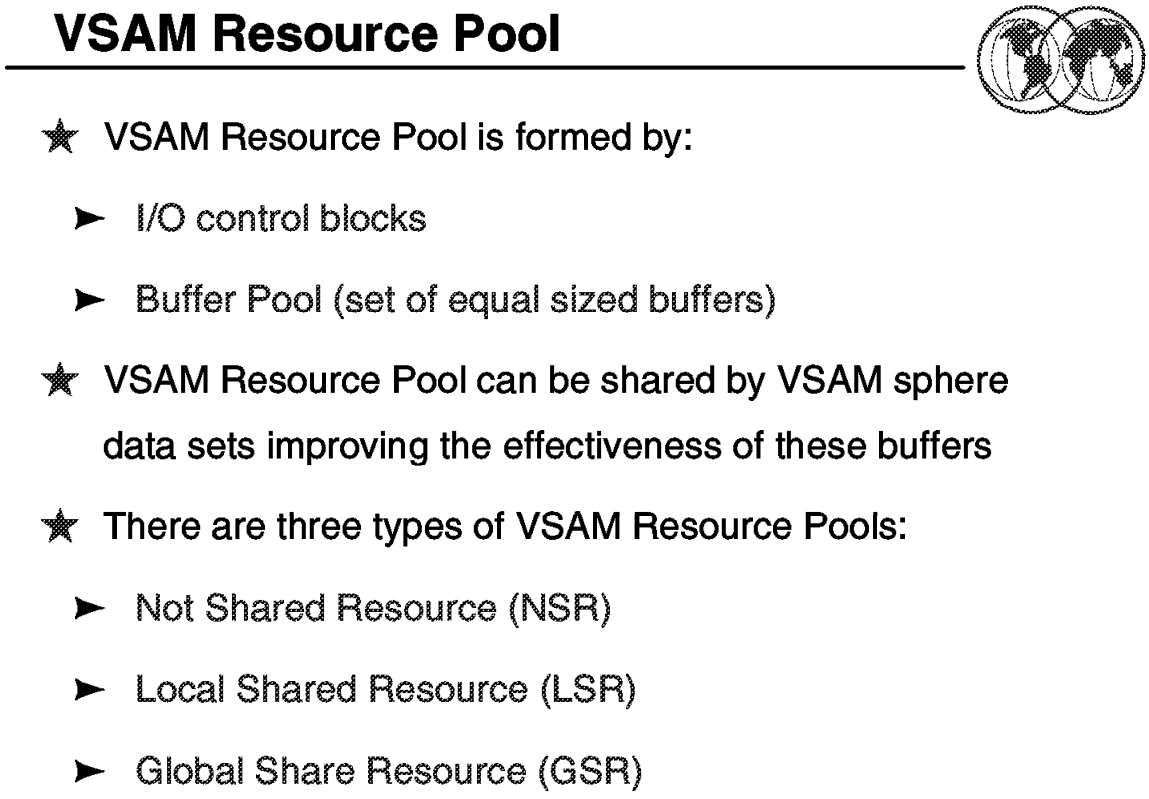


Figure 61. VSAM resource pool

1.19.1 VSAM resource pool

VSAM resource pool is a set of VSAM I/O control blocks plus a buffer pool. A buffer pool is a collection of same-sized I/O buffers plus control information describing the occupancy of such buffers. The objective of a buffer pool is to avoid I/O operations and consequently to improve performance.

For more efficient use of virtual storage, buffer pools can be shared among data sets (except linear data sets), using globally or locally shared buffer pools. There are three types of resource pools (depending on the type of the associated buffer pool):

- Not shared resource (NSR)
  - Implicitly constructed by OPEN.
  - Not shared among VSAM data sets.
  - Used by HLL.
  - Located in the private area.
  - Buffers are managed via a sequential algorithm. If you have a batch job accessing a VSAM file randomly, it is a good idea to implement BLSR (a component of the SmartBatch product). It transparently converts from NSR to LSR allowing a huge performance gain.
- Local shared resource (LSR)
  - Shared among VSAM data sets accessed by tasks in the same address space.

  - Located in the private area and ESO hiperspace. With hiperspace, VSAM buffers are located in expanded storage to improve the processing of VSAM data sets.
  - Explicitly constructed via macro BLDVRP.
  - Buffers are managed via LRU algorithm.
- Global shared resource (GSR)
  - Shared among VSAM data sets accessed by tasks is multiple address spaces.
  - Located in CSA.
  - Explicitly constructed via macro BLDVRP.
  - Buffers are managed via LRU algorithm.

These options are declared in the ACB macro of the VSAM data set (MACRF keyword).

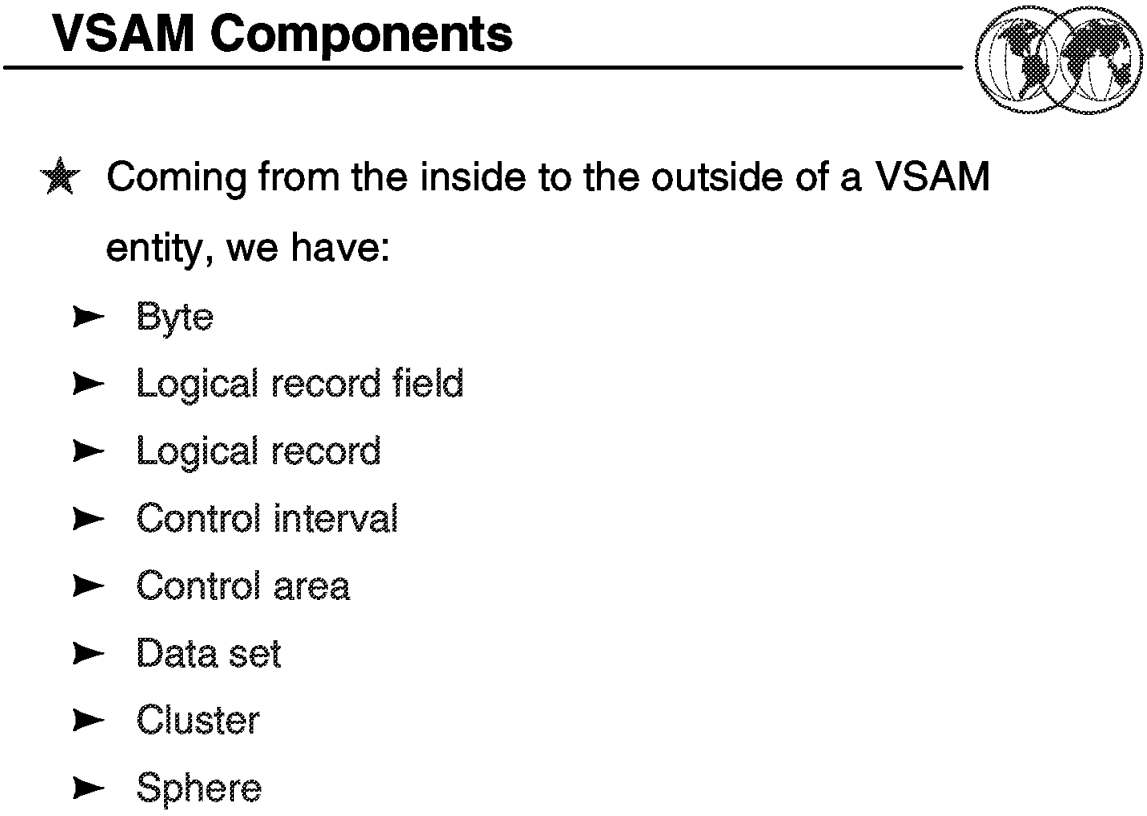


Figure 62. VSAM components

1.19.2 VSAM components

- A control interval (CI) is a unit of information that VSAM transfers between virtual storage and disk storage. The size of the CIs can vary from one data set to another, but all the CIs within the data portion for a particular data set must be of the same length. One CI can be made of one or several physical blocks.

  A CI consists of the following:

  - Logical records stored from beginning to end
  - Unused space - used to absorb inclusions
  - Control Information at the end formed by one record definition field (RDF) per logical record and one control interval definition field (CIDF) per CI

  In certain VSAM organizations there are free CIs, used also to absorb inclusions through the CI split mechanism.

- A control area (CA) is a group of CIs grouped together into fixed-length areas of DASD. Generally, a CA has the size of one physical DASD cylinder. Its major objective is to absorb inclusions (not serviced by the CI split) through the CA split mechanism.
- A VSAM data set is composed of one or more CAs. Then, its size is a multiple of the control area size.
- A cluster is the combination of the data component (data set) and the index component (data set) for a KSDS. The cluster provides a way to treat index and data components as a single component with its own name. You can also give each component (data set) a name. Fixed-length RRDSs,

entry-sequence data sets, and LDS are considered to be clusters without index components. To be consistent, they are given cluster names that are normally used when processing the data set.

- Sphere is a VSAM cluster and its associated data sets. These data sets are alternate index (AIX) of the cluster. An AIX is a key-sequenced data set containing index entries organized by the alternate keys of its associated base data records. It provides another way of locating records in the data component of a cluster.

  An AIX can be defined over a key-sequenced or entry-sequenced cluster only.

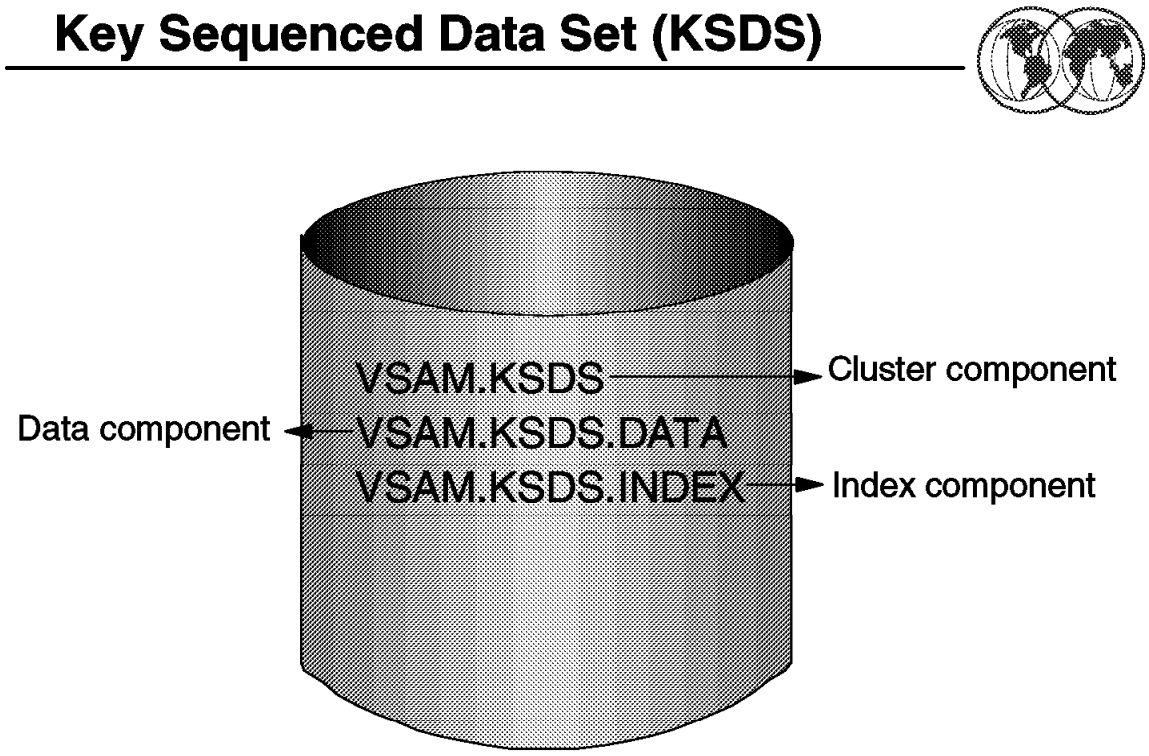


Figure 63. Key sequenced data set (KSDS)

1.19.3 Key sequenced data set (KSDS)

In this visual we start to view each one of the VSAM organizations.

In a KSDS, logical records are placed in the data set in ascending collating sequence by key. The key contains a unique value, which determines the record′s collating position in the data set. The key must be in the same position in each record.

The key data must be contiguous and each key must be unique. After it is specified, the value of the key cannot be altered, but the entire record may be deleted.

When a new record is added to the data set, it is inserted in its collating sequence by key.

A KSDS has a data and an index component. The index component keeps track of the used keys and is used by VSAM to retrieve quickly a record from the data component when a request is made for a record with a certain key.

A KSDS can have fixed or variable length records.

A KSDS can be accessed in either sequential, direct, or skip sequential (you process sequentially, but skipping some portions of the data set) mode.

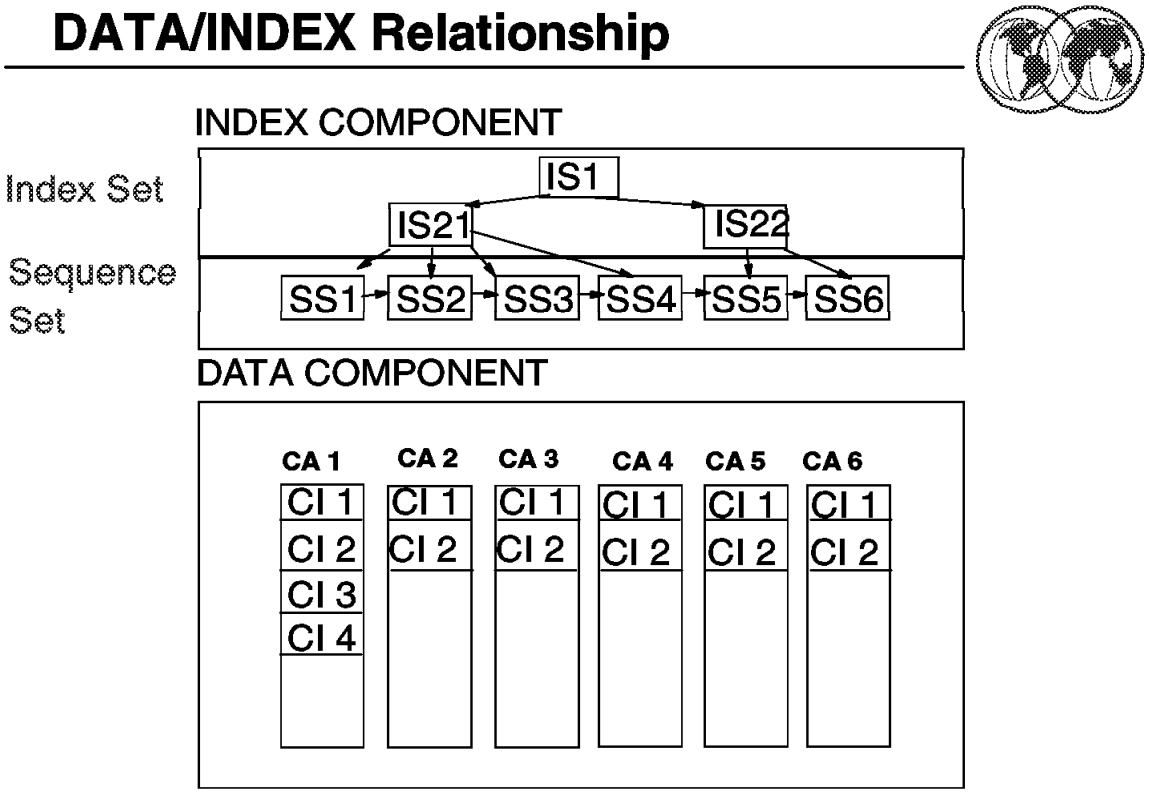


Figure 64. Data/Index relationship

1.19.4 Data/Index relationship

A KSDS has an index that relates key values to the relative locations in the data set. This index is called the prime index. It has two uses:

- Locate the collating position when inserting records
- Locate records for retrieval

When initially loading a data set, records must be presented to VSAM in a key sequence. The index is built automatically by VSAM as the data set is loaded with records. When a data CI is loaded with records, VSAM makes an entry in the index. The entry consists of the highest possible key in the data CI and a pointer to the beginning of that CI.

A VSAM index can consist of more than one index level. Each level contains a set of records with entries giving the location of the records in the next lowest level.

- *Sequence set* contains the index CIs at the lowest level. There is one CI in the sequence set for each data CA. It contains pointers and high key information for each CI within the data CA. It contains also horizontal pointers from one sequence set CI to the next (higher keyed) sequence set CI.
- *Index set* contains the remainder of the index component. If there is more than one sequence-set level record, VSAM automatically builds another index level. Each CI in the index set contains pointers and high key information for CIs in the next lower level of the index.

The highest level of the index always contains a single index CI.

In this visual SS1 is a sequence set CI located in the index. It has four records, the:

- First has the highest key and location of the data CI 1
- Second has the highest key and location of the data CI 2
- Third has the highest key and location of the data CI 3
- Fourth has the highest key and location of the data CI 4

Where:

IS Index Set

SS

Sequence Set

CA Control Area

All four CIs are located in the data CA1.

SS2 has the same type of information for the data CA2.

SSn sequence set records are also clustered in CAs. There is one IS2n (index set level two) for each one of the CAs. Each record of the IS2n points to the highest key and location of one SSn located in the SSn CA pointed to by this IS2n.

The same logic applies to the higher levels of indexes. For each new CA below, a new CI is created above, until just having only one IS1 (index set level one).

1.19.4.1 Requests for data

Request 1: A control interval from CA2 is requested by string 1.

- The highest level index set, IS1, is read into an index buffer. IS1 remains in this buffer for all requests.
- IS1 points to IS2, which is read into a second index buffer.
- IS2 points to the sequence set, SS2, which is read into an index buffer for string 1.
- SS2 points to a control interval in CA2. This control interval is read into a data buffer for string 1.

Request 2: A control interval from CA3 is requested by string 2.

- IS1 and IS2 remain in their respective buffers.
- SS3 is read into an index buffer for string 2.
- SS3 points to a control interval in CA3. This control interval is read into a data buffer for string 2.

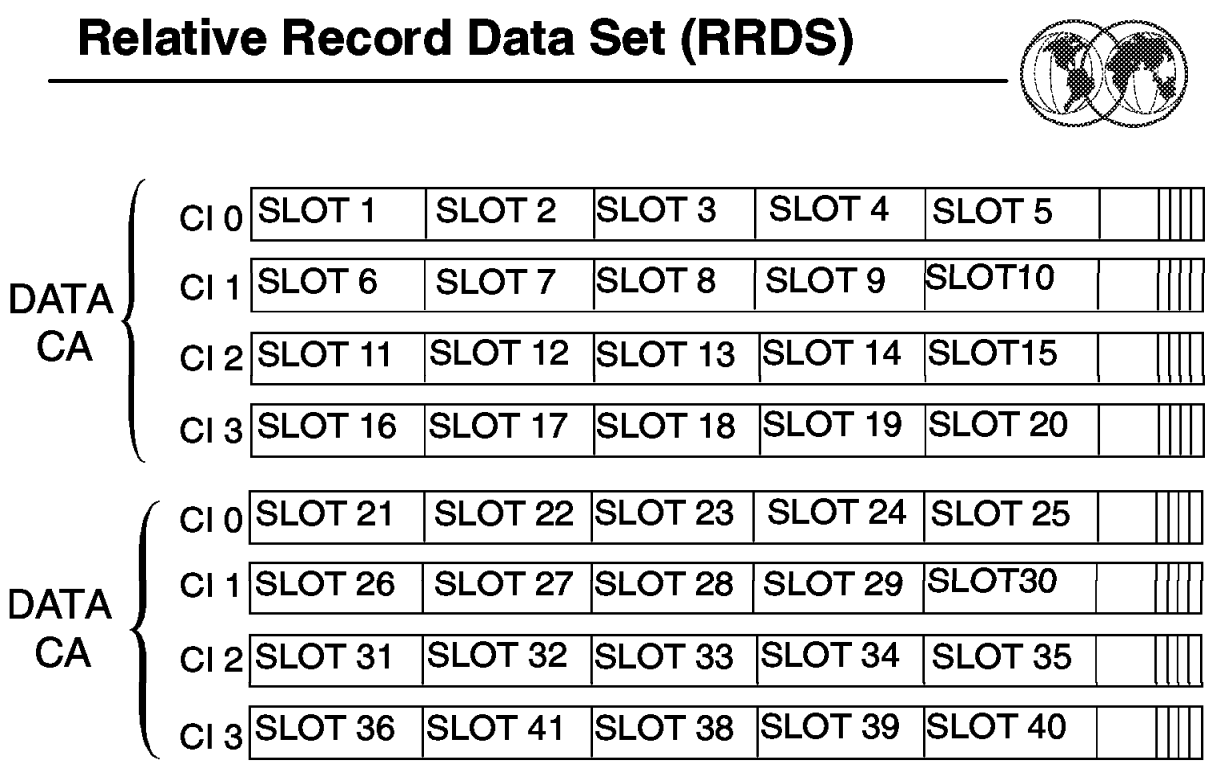


Figure 65. Relative record data set (RRDS)

1.19.5 Relative record data set (RRDS)

A relative record data set (RRDS) consists of a number of preformatted fixed-length slots. Each slot has a unique relative record number, and the slots are sequenced by ascending relative record number. Each (fixed length) record occupies a slot, and is stored and retrieved by the relative record number of that slot. The position of a data record is fixed, its relative record number cannot change.

An RRDS has a data component only.

Random load of an RRDS requires a user program.

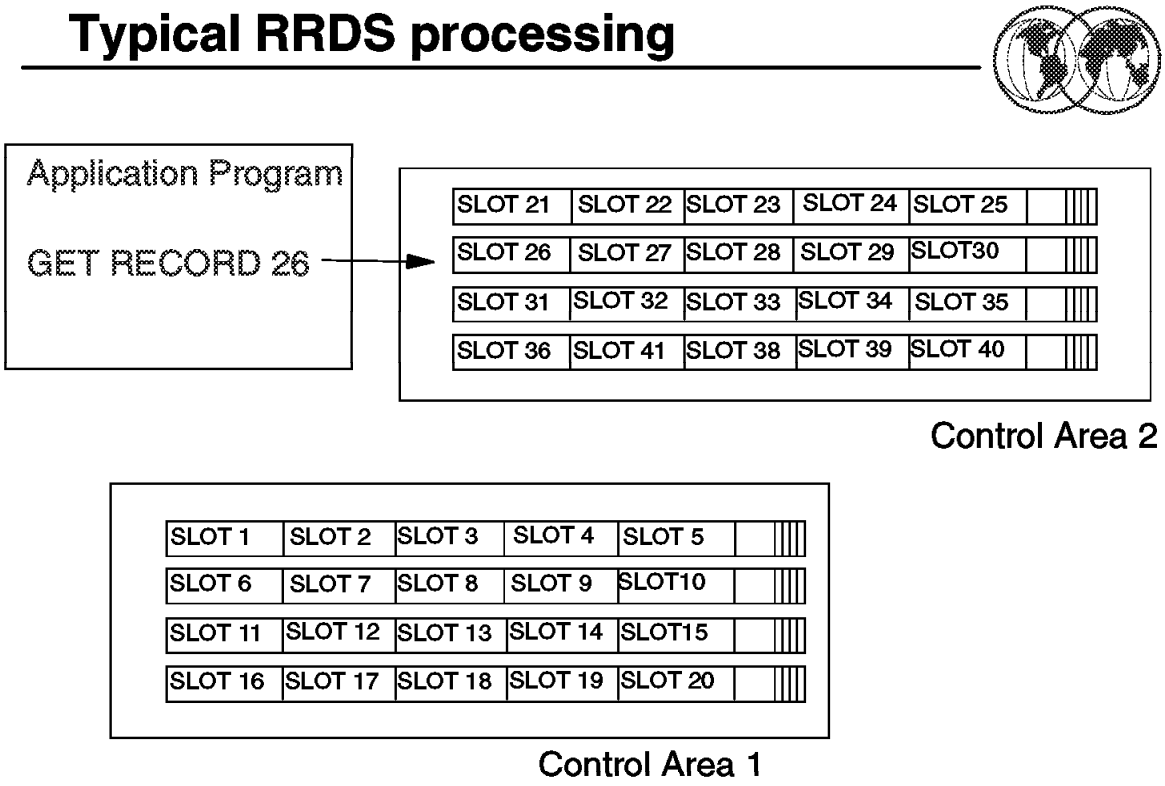


Figure 66. Typical RRDS processing

1.19.6 Typical RRDS processing

The application program inputs the relative record number of the target record and VSAM is able to find very fast its location using a formula that takes into consideration the geometry of the DASD device. The relative number is always used as a search argument. For an RRDS, three types of processing are supported:

- Sequential processing.
- Skip-sequential processing.
- Direct processing. In this case the randomization routine is supported by the application program.

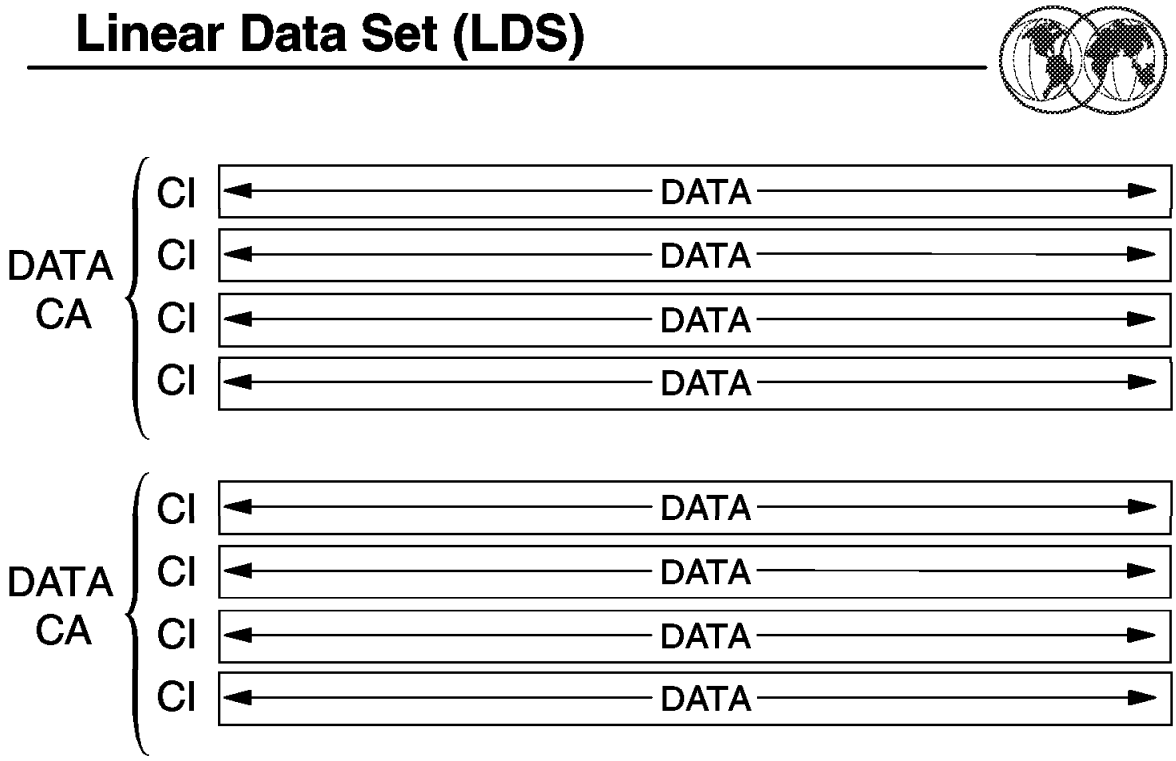


Figure 67. Linear data set (LDS)

1.19.7 Linear data set (LDS)

A linear data set is a VSAM data set with a CI size of 4096 bytes. An LDS has no imbedded control information in its CI, that is, no RDFs and CIDFs. So, all LDS bytes are *data* bytes. Logical records must be blocked and deblocked by the application program, but logical records do not exist from the point of view of VSAM.

IDCAMS is used to define a linear data set. An LDS has only a data component. An LDS data set is just a physical sequential VSAM data set made of 4 KB blocks, but with a revolutionary buffer technique called data-in-virtual (DIV).

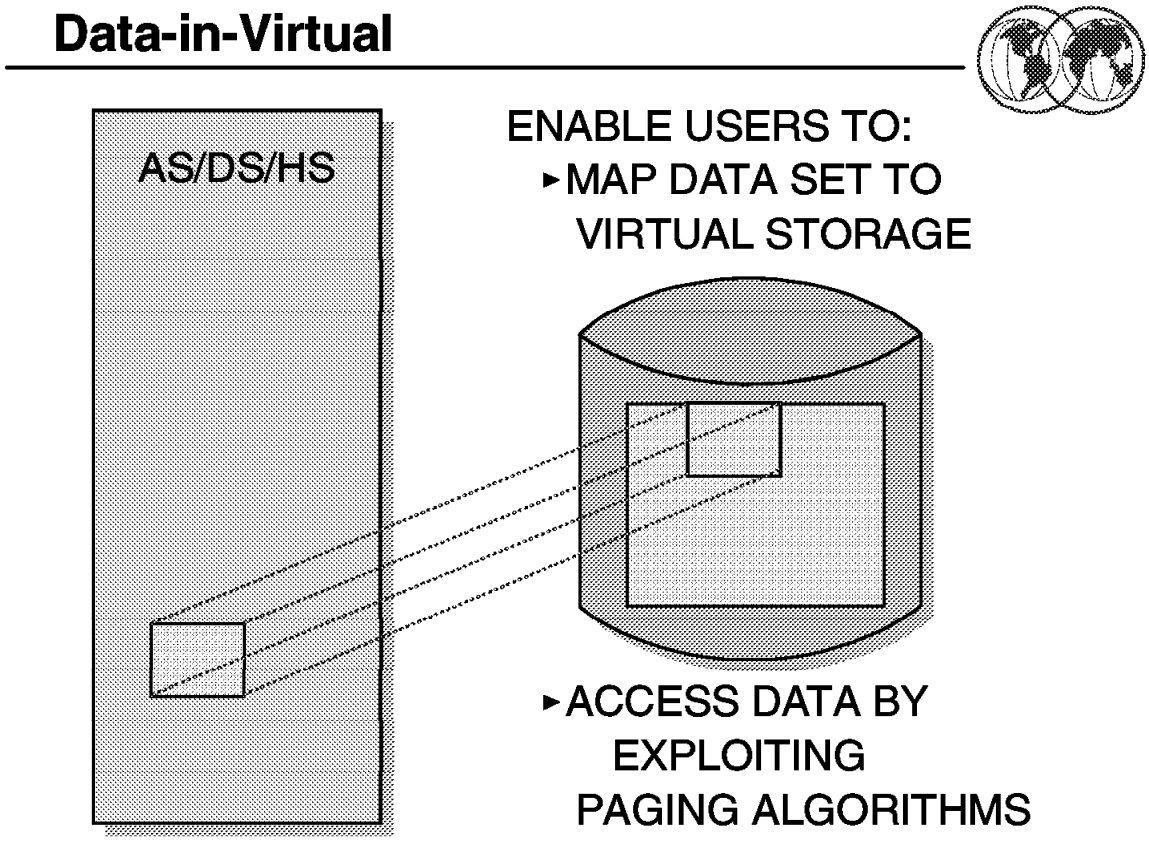


Figure 68. Data-in-virtual

1.19.8 Data-in-virtual

Application programs can use data-in-virtual (DIV) to *map* a data set or a portion of a data set into an address space, a data space, or a hiperspace.

Data is read into central storage via the paging algorithms only when that block is actually referenced. During RSM page-steal processing, only changed pages are written to auxiliary storage. Unchanged pages are discarded since they can be retrieved again from the permanent data set.

DIV is designed to improve the performance of applications that process large files nonsequentially and process them with significant locality of reference. It reduces the number of I/O operations that are traditionally associated with data retrieval. Likely candidates are large arrays or table files.

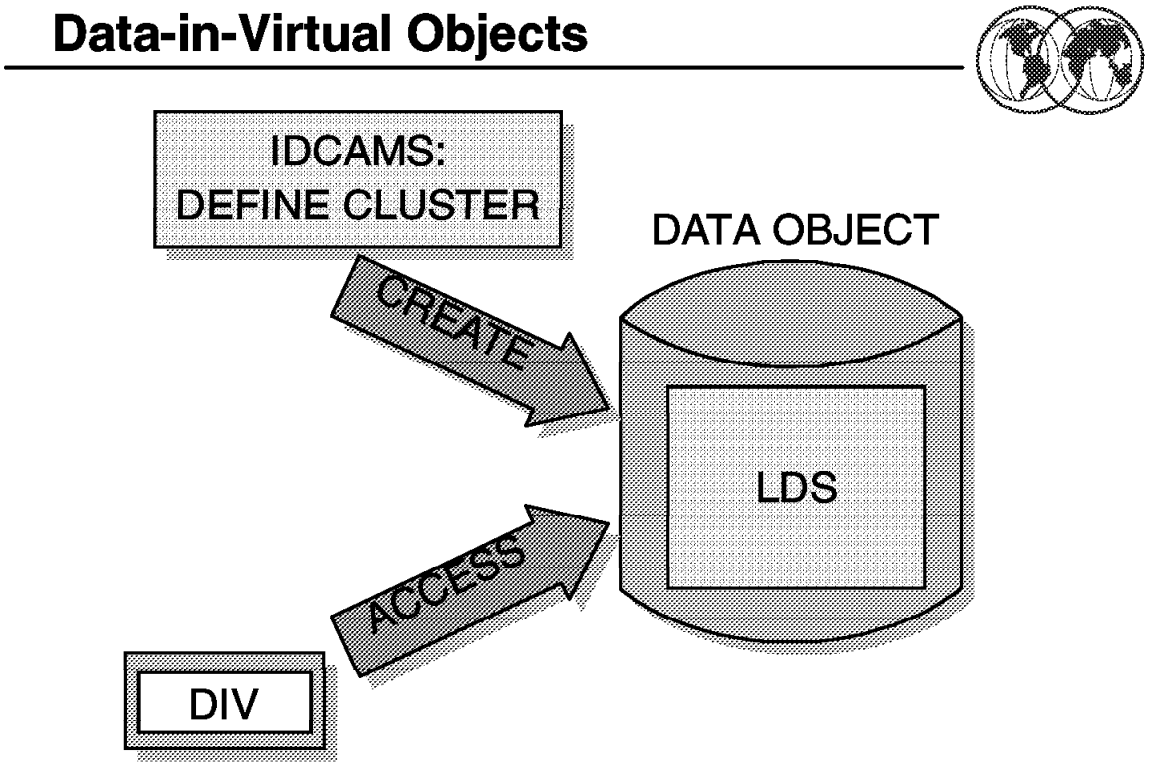


Figure 69. Data-in-virtual objects

1.19.9 Data-in-virtual objects

A linear data set is a VSAM data set with a control interval size of 4096 bytes to 32,768 bytes in increments of 4096 bytes. A linear data set does not have imbedded control information. All linear data set bytes are data bytes. Only integrated catalog facility catalogs can support a linear data set.

A linear data set is processed as an entry-sequenced data set, with certain restrictions. Because a linear data set does not contain control information (CIDFs and RDFs), it cannot be accessed as if it contained individual records. You can access a linear data set with the DIV macro. If using DIV to access the data set, the control interval size must be 4096 otherwise the data set will not be processed.

For information on how to use data-in-virtual (DIV), see *OS/390 MVS Programming: Assembler Services Guide*, GC28-1762.

When a linear data set is accessed with the DIV macro, it is referred to as the data-in-virtual object or the data object.

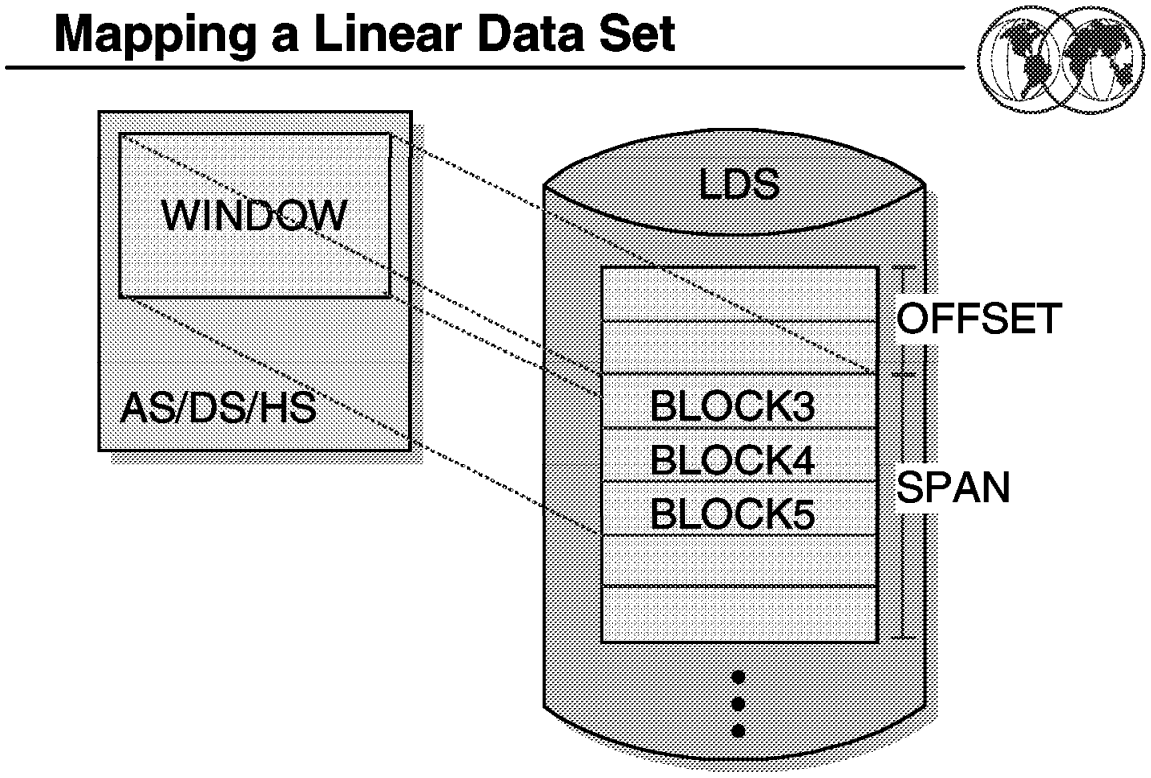


Figure 70. Mapping a linear data set

1.19.10 Mapping a linear data set

To establish a map from a linear data set to a window (a program-provided area in multiples of 4 KB on a 4 KB boundary), the program issues:

- DIV IDENTIFY to introduce (allocate) a linear data set to data-in-virtual services.
- DIV ACCESS to cause a VSAM open for the data set and indicate access mode (read/update).
- DIV MAP to enable the viewing of the data object by establishing an association between a program-provided area and the data object. The area may be in an address space, data space, or hiperspace.

No actual I/O is done until the program references the data in the window. The reference will result in a page fault which causes data-in-virtual services to read the data from the linear data set into the window.

DIV SAVE can be used to write out changes to the data object. DIV RESET can be used to discard changes made in the window since the last SAVE operation.

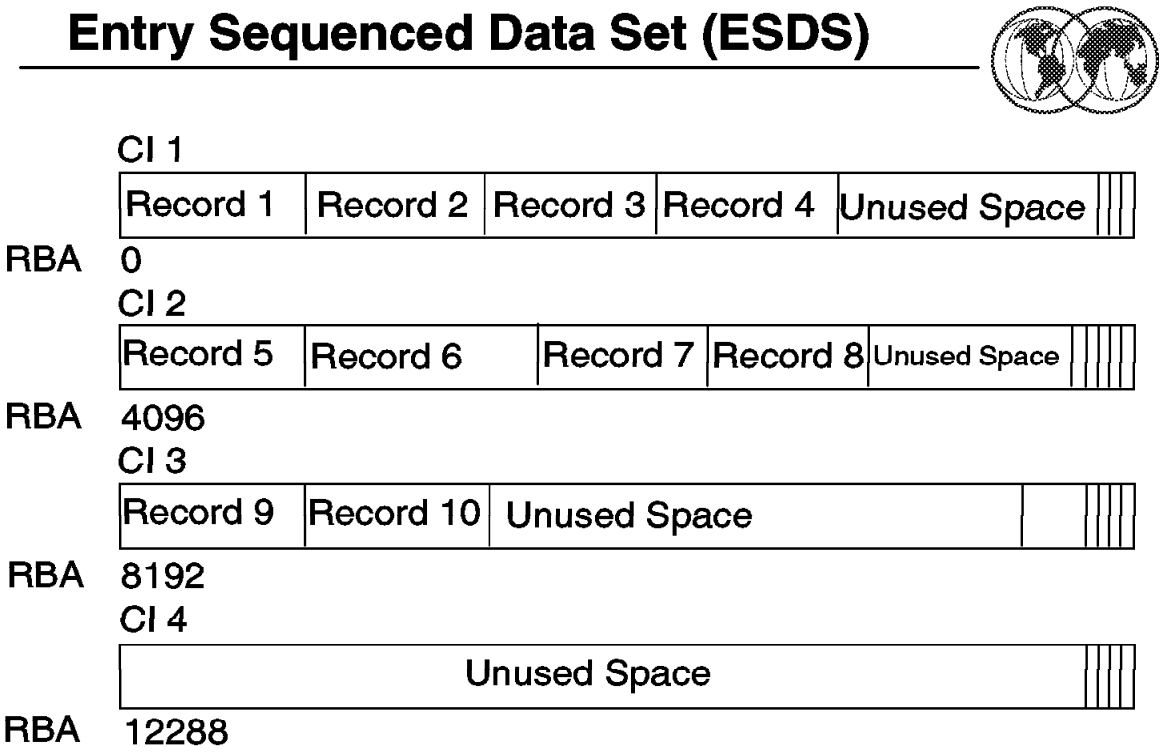


Figure 71. Entry sequenced data set (ESDS)

1.19.11 Entry sequenced data set (ESDS)

An ESDS is comparable with a sequential data set. It contains fixed or variable-length records. Records are sequenced by the order of their entry in the data set, rather than by a key field in the logical record. All new records are placed at the end of the data set. An ESDS has only a data component.

Records can be accessed sequentially or by relative byte address (RBA). When a record is loaded or added, VSAM indicates its relative byte address (RBA). The RBA is the offset of this logical record from the beginning of the data set. The first record in a data set has an RBA of 0; the second record has an RBA equal to the length of the first record, and so on. The RBA of a logical record depends only on the record's position in the sequence of records. The RBA is always expressed as a fullword binary integer.

Although an entry-sequenced data set does not contain an index component, alternate indexes are allowed. You can build an alternate index to keep track of these RBAs.

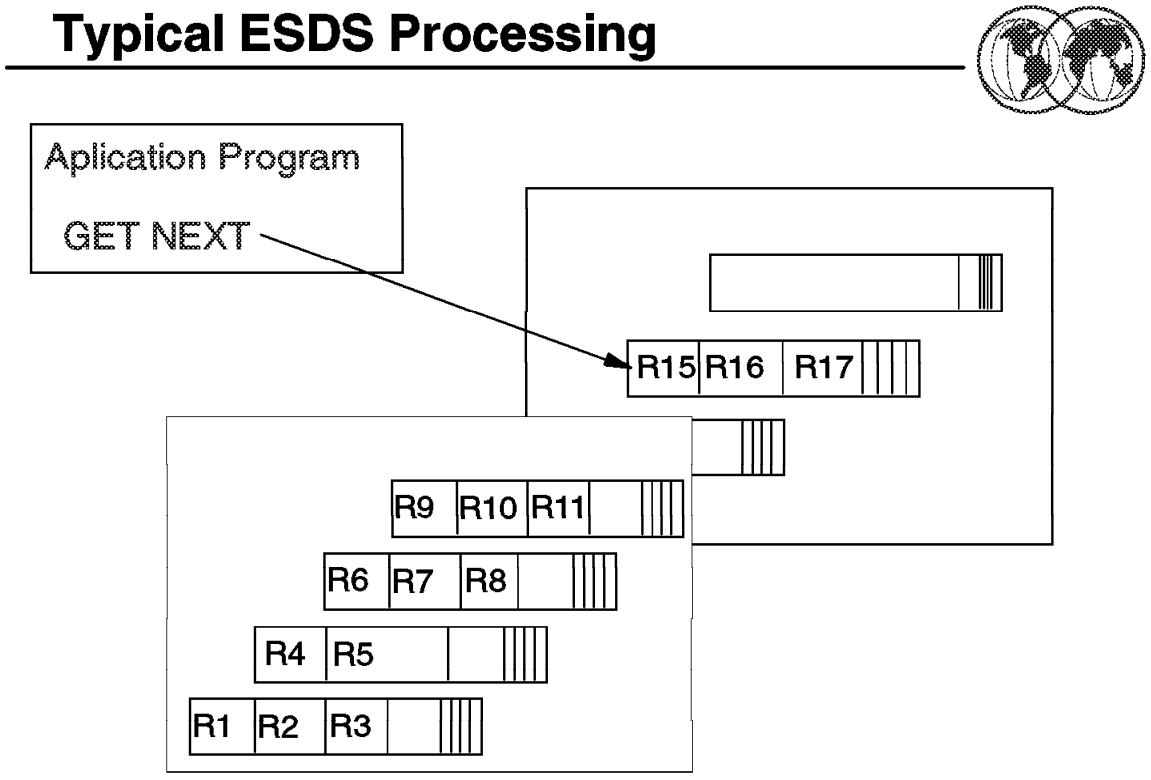


Figure 72. Typical ESDS processing (ESDS)

1.19.12 Typical ESDS processing (ESDS)

For an ESDS, two types of processing are supported:

- Sequential access is most common
- Direct (or random) access requires the program to give the RBA of the record

Skip sequential is not allowed.

Existing records can never be deleted. If the application wants to delete a record, it must flag that record as inactive. As far as VSAM is concerned, the record is not deleted. They can be updated, but without length change.

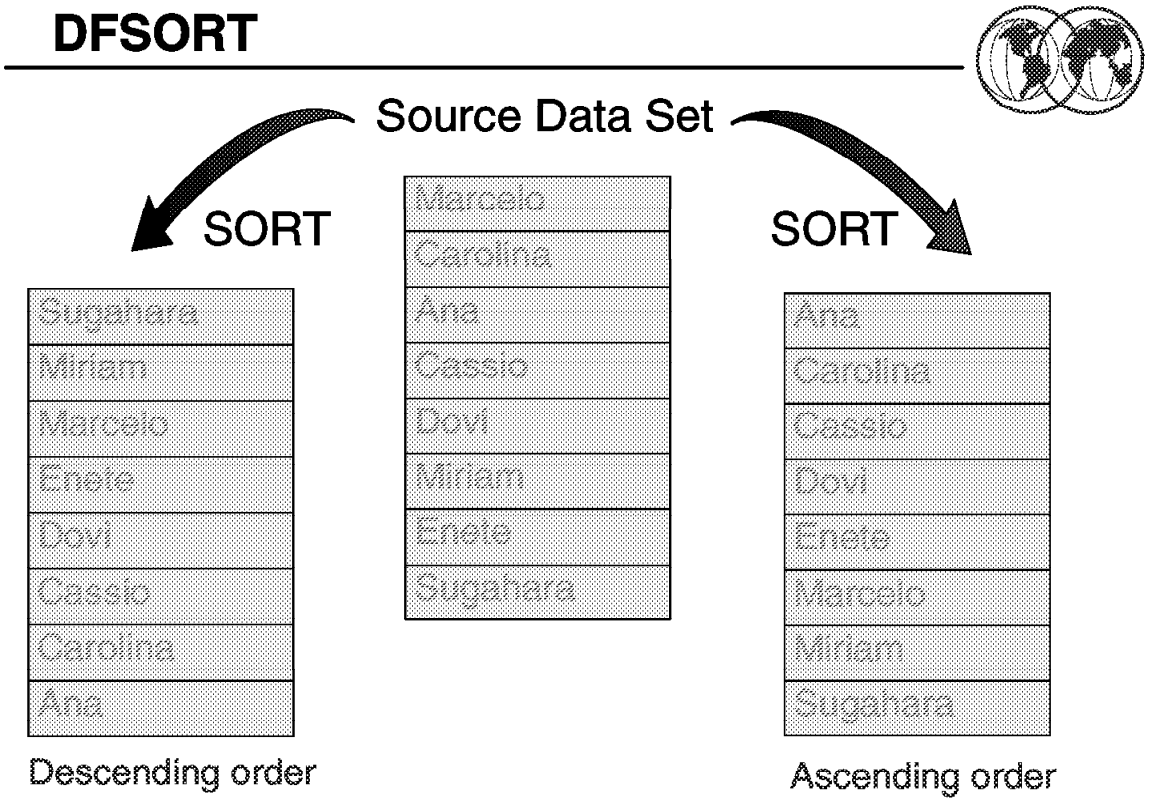


Figure 73. DFSORT

1.20 DFSORT

In this visual we start to present the products included in DFSMS/MVS umbrella.

The DFSORT licensed program is a high-performance data arranger for OS/390 users.

With DFSORT, you can sort, merge, and copy data sets using EBCDIC, S/390 decimal or binary keys.

DFSORT merges data sets by combining two or more files of sorted records to form a single data set of sorted records.

You can use DFSORT to do simple tasks such as alphabetizing a list of names, or you can use it to aid complex tasks such as taking inventory or running a billing system. You can also use DFSORT′s record-level editing capability to perform data-management tasks.

For most of the processing done by DFSORT, the whole data set is affected. However, some forms of DFSORT processing involve only certain individual records in that data set.

While sorting, merging, or copying data sets, you can also:

- Select a subset of records from an input data set. You can include or omit records that meet specified criteria. For example, when sorting an input data set containing records of course books from many different school departments, you can sort the books for only one department.

- Reformat records, add or delete fields, and insert blanks, constants, or binary zeros. For example, you can create an output data set that contains only certain fields from the input data set arranged differently.
- Sum the values in selected records while sorting or merging (but not while copying). In the example of a data set containing records of course books, you can use DFSORT to add up the dollar amounts of books for one school department.
- Create multiple output data sets and reports from a single pass over an input data set. For example, you can create a different output data set for the records of each department.
- Sort, merge, include, or omit records according to the collating rules defined in a selected local.
- Alter the collating sequence when sorting or merging records (but not while copying). For example, you can have the lowercase letters collate after the uppercase letters.
- Sort, merge, or copy Japanese data if the IBM Double Byte Character Set Ordering Support (DBCS Ordering) (5665-360 Licensed Program, Release 2.0 or an equivalent product) is used with DFSORT to process the records.

DFSORT has utilities such as ICETOOL which is a multipurpose DFSORT utility that uses the capabilities of DFSORT to perform multiple operations on one or more data sets in a single step.

For articles, online books, news, tips, techniques, examples, and more, visit the DFSORT/MVS home page at URL:

`http://www.ibm.com/storage/dfsort/`

For further information about DFSORT, refer to *DFSORT Getting Started with DFSORT R14*, SC26-4109, and other DFSORT books.

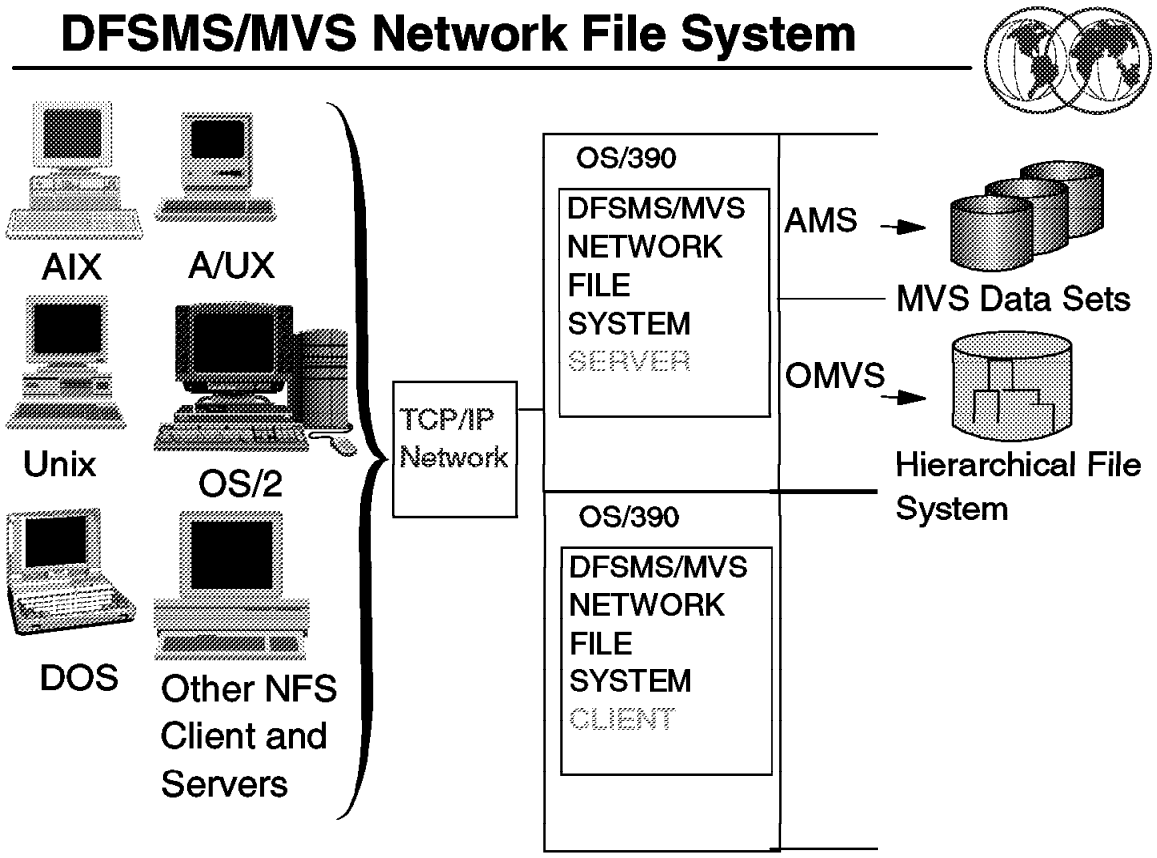


Figure 74. DFSMS/MVS Network File System

1.21 DFSMS/MVS Network File System

A client is a computer or process that requests services on the network. A server is a computer or process that responds to a request for service from a client. A user accesses a service, which allows the use of data or other resources.

This visual illustrates the client-server relationship. The upper center portion of the figure shows the DFSMS/MVS NFS address space server. The lower right portion of the figure shows the DFSMS/MVS NFS address space client. The left portion of the figure shows various NFS clients and servers which can interact with the DFSMS/MVS NFS server and client. The center of the figure shows the Transmission Control Protocol/Internet Protocol (TCP/IP) network used to communicate between clients and servers.

With the DFSMS/MVS NFS server, you can remotely access MVS/ESA conventional data sets or UNIX server MVS files from workstations, personal computers, and other systems that run client software for the Sun NFS Version 2 protocols on a TCP/IP network. The DFSMS/MVS NFS server acts as an intermediary to read, write, create, or delete UNIX server MVS files and MVS data sets that are maintained on an MVS host system. The remote MVS data sets or UNIX server MVS files are mounted from the host processor to appear as local directories and files on the client system. This server makes the strengths of an MVS host processor—storage management, high-performance disk storage, security, and centralized data—available to the client platforms.

With the DFSMS/MVS NFS client you can allow basic sequential access method (BSAM), queued sequential access method (QSAM), virtual storage access method (VSAM), and UNIX server MVS users

and applications transparent access to data on systems which support the Sun NFS Version 2 protocols. The remote NFS server can be an MVS, UNIX**, AIX, OS2, or other system. The DFSMS/MVS NFS client is implemented on UNIX server MVS and implements the client portion of the Sun NFS Version 2 protocols.

The Network File System, then, can be used for:

- File sharing between platforms
- File serving (as a data repository)

For further information about NFS, refer to *DFSMS/MVS Network File System Customization and Operation*, SC26-7029, and *DFSMS/MVS Network File System User's Guide*, SC26-7028.

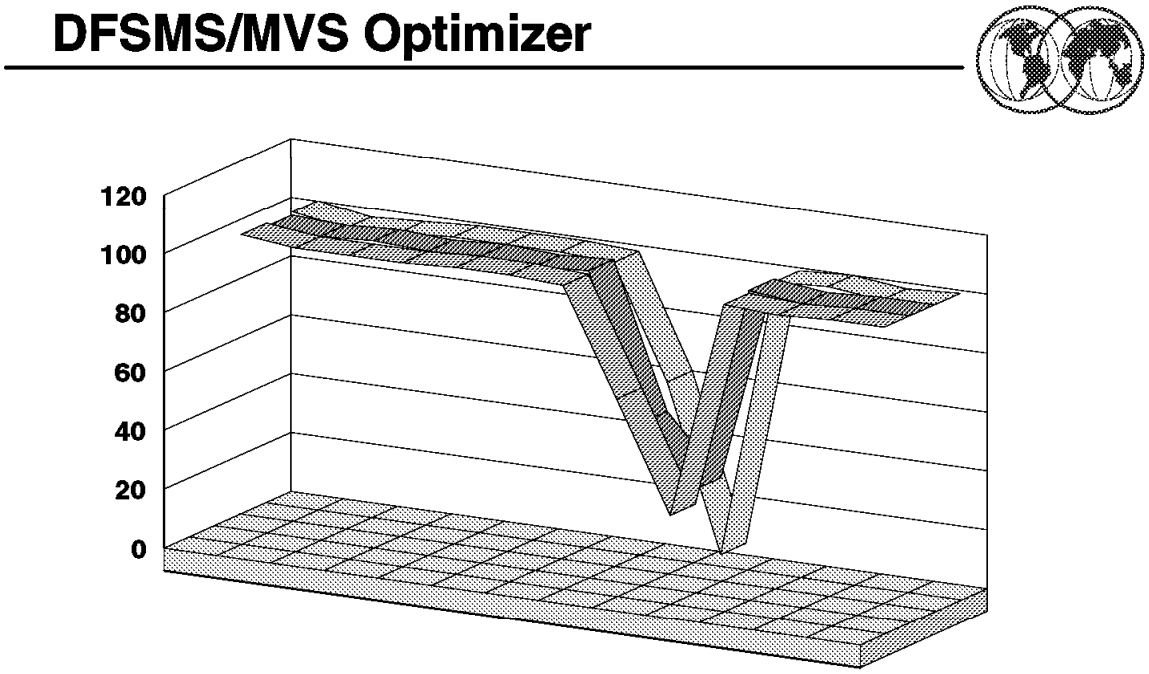


Figure 75. DFSMS/MVS Optimizer

1.22 DFSMS/MVS Optimizer

The DFSMS Optimizer provides analysis and simulation information for both SMS and non-SMS users. The DFSMS Optimizer can help you maximize storage use and minimize storage costs. It provides methods and facilities for you to:

- Monitor and tune DFSMShsm functions as migration and backup
- Create and maintain a historical database of system and data activity
- Fine tune an SMS configuration, by performing in-depth analysis of:
  - Management class policies, including simulations and cost-benefit-analysis using your storage component costs
  - Storage class policies for SMS data, with recommendations for both SMS and non-SMS data
  - High I/O activity data sets, including recommendations for placement and simulation for cache and expanded storage
  - Storage hardware performance of subsystems and volumes including I/O rate, response time, and caching statistics
- Simulate potential policy changes and understand the costs of those changes
- Produce presentation-quality charts

For more information on the DFSMS Optimizer, see the *DFSMS Optimizer User's Guide and Reference*, SC26-7047.

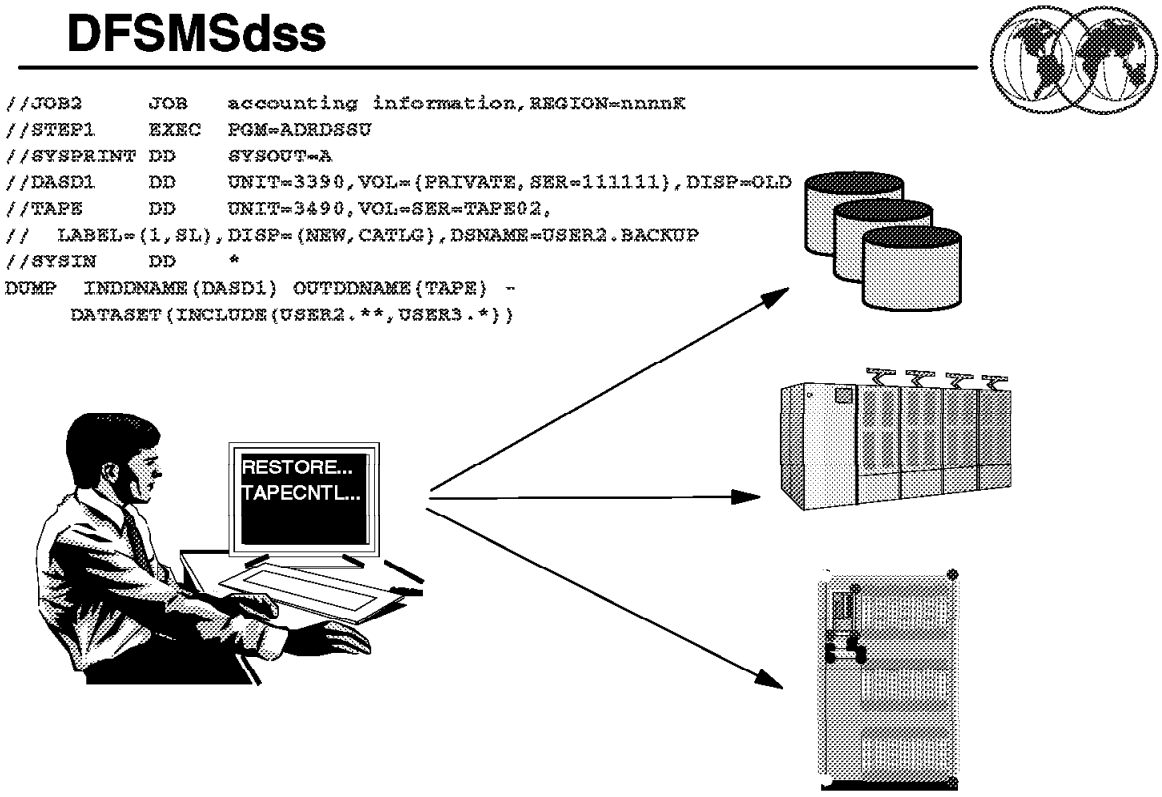


Figure 76. DFSMSdss

1.23 DFSMSdss

DFSMSdss is a direct access storage device (DASD) data and space management tool. DFSMSdss works on DASD volumes only in the MVS environment. You can use DFSMSdss to:

- Copy and move data sets between volumes of like and unlike device types

  Note: Like devices have the same track capacity and number of tracks per cylinder (for example, 3380 Model D, Model E, and Model K). Unlike DASD devices have different track capacities (for example, 3380 and 3390), a different number of tracks per cylinder, or both.
- Dump and restore data sets, entire volumes, or specific tracks
- Convert data sets and volumes to and from SMS management
- Compress partitioned data sets
- Release unused space in data sets
- Reduce or eliminate DASD free-space fragmentation by consolidating free space on a volume
- Implement concurrent copy in 9390/3990 control units. If the control unit is a 9393 RVA, a snapshot is transparently generated without any change in the JCL.

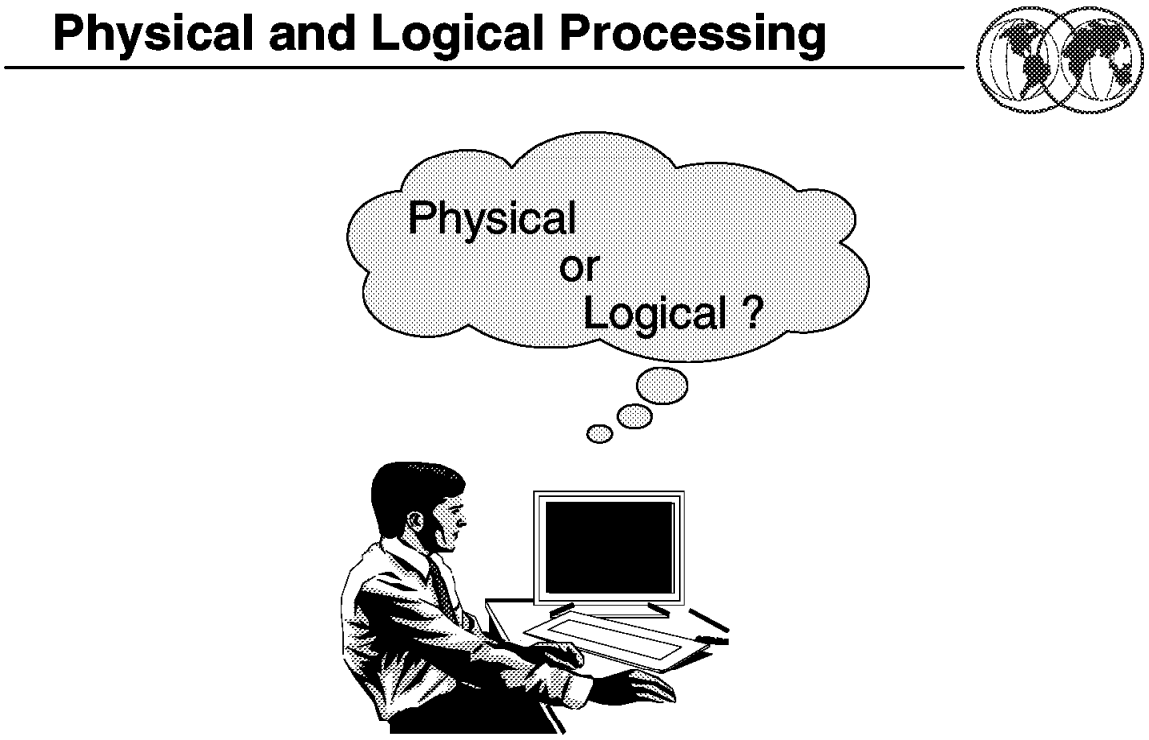


Figure 77. DFSMSdss: physical and logical processing

1.23.1 DFSMSdss: physical and logical processing

Before you begin using DFSMSdss, you should understand the difference between logical and physical processing. DFSMSdss can perform two kinds of processing when executing COPY, DUMP, and RESTORE commands:

- Logical processing operates against data sets independently of physical device format.
- Physical processing moves data at the track-image level and operates against volumes, tracks, and data sets.

Each type of processing offers different capabilities and advantages.

During a restore operation, the data is processed the same way it is dumped because physical and logical dump tapes have different formats. If a data set is dumped logically, it is restored logically; if it is dumped physically, it is restored physically. A data set restore operation from a full volume dump is a physical data set restore operation.

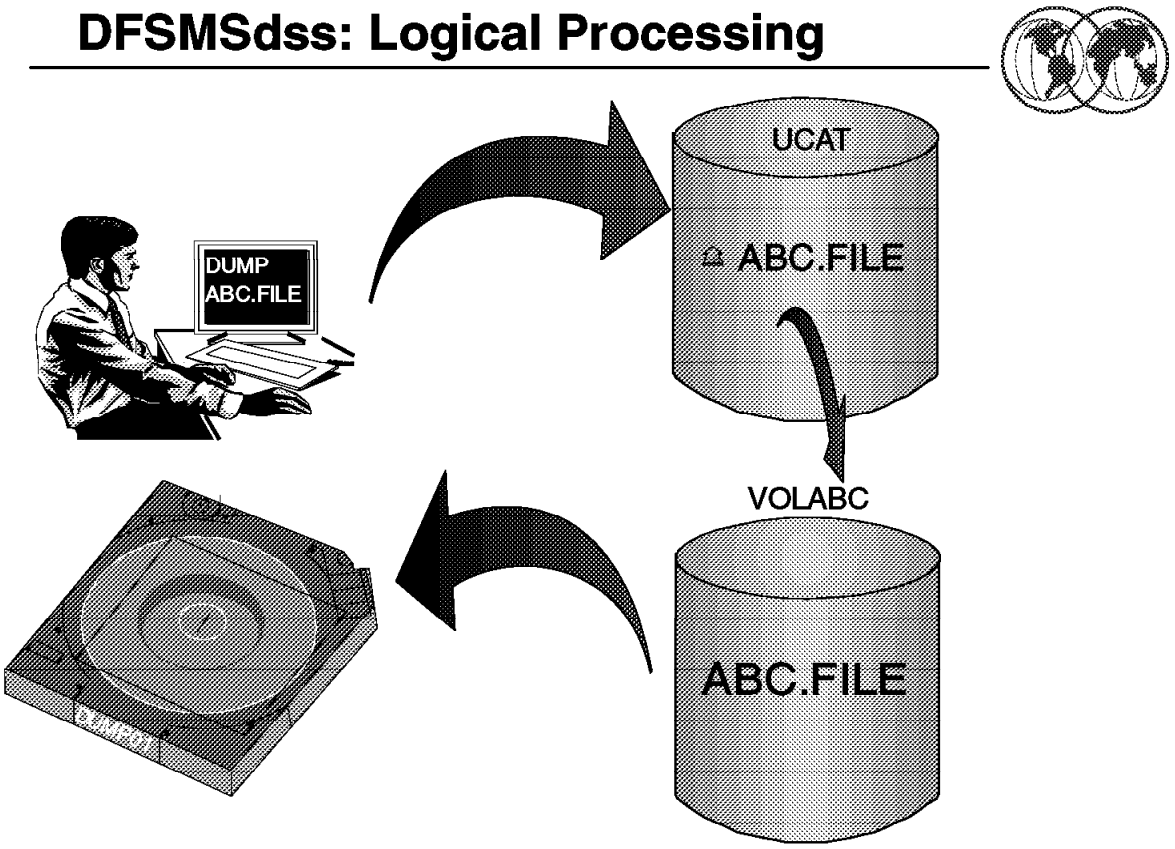


Figure 78. DFSMSdss: logical processing

1.23.2 DFSMSdss: logical processing

A logical copy, dump, or restore operation treats each data set and its associated information as a logical entity, and processes an entire data set before beginning the next one.

Each data set is moved by tracks from the source device and is potentially written to the target device as a set of data records, allowing data movement between devices with different track and cylinder configurations. Checking of data record consistency is not performed during dump operation.

DFSMSdss performs logical processing if:

- You specify the data set keyword with the COPY command. A data set copy is always a logical operation, regardless of how or whether you specify input volumes.
- You specify the data set keyword with the DUMP command, and either no input volume is specified, or LOGINDDNAME or LOGINDYNAM is used to specify input volumes.
- The RESTORE command is performed, and the input volume was created by a logical dump.

Catalogs and VTOCs are used to select data sets for logical processing. If you do not specify input volumes, the catalogs are used to select data sets for copy and dump operations.

1.23.2.1 When to use logical processing

Use logical processing for the following situations:

- Data is copied to an unlike device type.
- Logical processing is the only way to move data between unlike device types.
- Data that may need to be restored to an unlike device is dumped.
- Data must be restored the same way it is dumped. This is particularly important to bear in mind when making backups that you plan to retain for a long period of time (such as vital records backups). If a backup is retained for a long period of time, it is possible that the device type it originally resided on will no longer be in use at your site when you want to restore it. This means you will have to restore it to an unlike device, which can be done only if the backup has been made logically.
- Aliases of VSAM user catalogs are to be preserved during copy and restore functions. Aliases are not preserved for physical processing.
- Unmovable data sets or data sets with absolute track allocation are moved to different locations.
- Multivolume data sets are processed.
- VSAM and multivolume data sets are to be cataloged as part of DFSMSdss processing.
- Data sets are to be deleted from the source volume after a successful dump or copy operation.
- Both non-VSAM and VSAM data sets are to be renamed after a successful copy or restore operation.
- You want to control the percentage of space allocated on each of the output volumes for copy and restore operations.
- You want to copy and convert a PDS to a PDSE or vice versa.
- You want to copy or restore a data set with an undefined DSORG to an unlike device.
- You want to keep together all parts of a VSAM sphere.

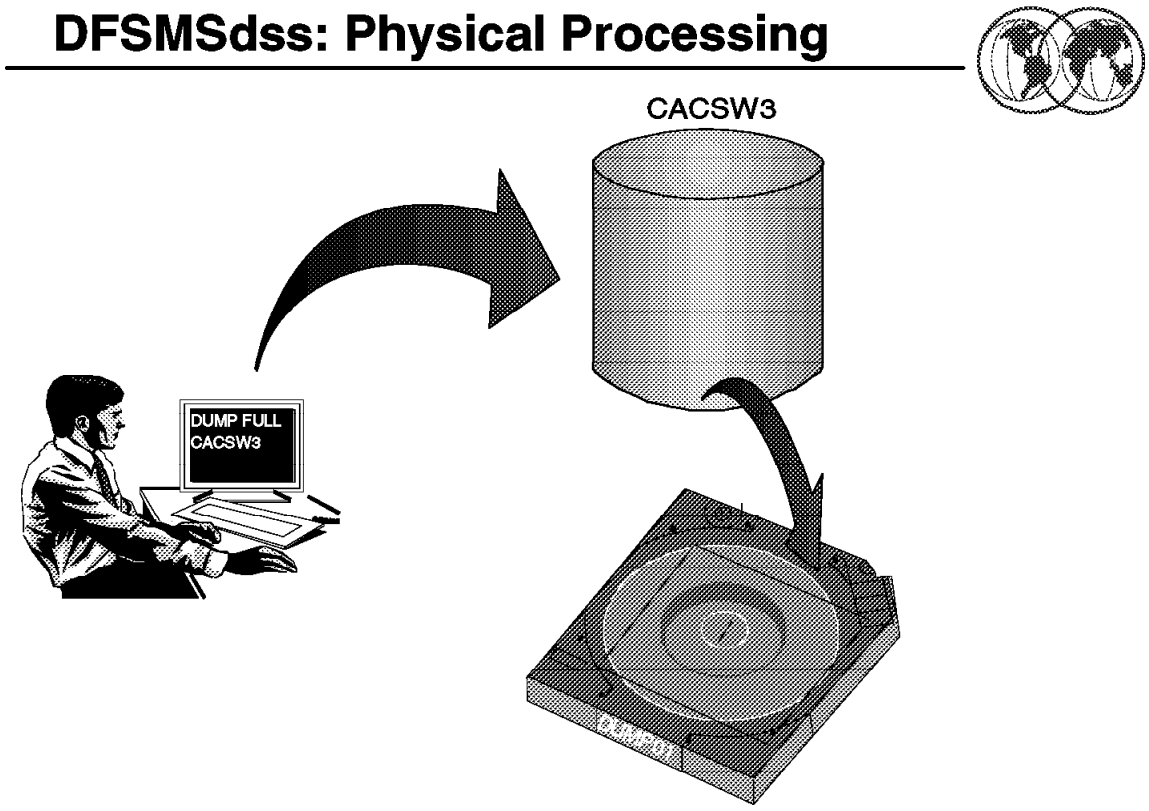


Figure 79. DFSMSdss: physical processing

1.23.3 DFSMSdss: physical processing

Physical processing moves data based on physical track images. Because data movement is carried out at the track level, only target devices with track sizes equal to those of the source device are supported. Physical processing operates on volumes, ranges of tracks, or data sets. For data sets, it relies only on volume information (in the VTOC and VVDS) for data set selection, and processes only that part of a data set residing on the specified input volumes.

DFSMSdss performs physical processing if:

- You specify the FULL or TRACKS keyword with the COPY or DUMP command. This results in a physical volume or physical tracks operation.

  Attention: Be aware that, when invoking the TRACKS keyword with the COPY and RESTORE commands, the TRACKS keyword should be used only for a data recovery operation. For example, you can use it to *repair* a bad track in the VTOC or a data set, or to retrieve data from a damaged data set. You cannot use it in place of a full-volume or a logical data set operation. Doing so could destroy a volume or impair data integrity.

- You specify the data set keyword on the DUMP command and input volumes with the INDDNAME or INDYNAM parameter. This produces a physical data set dump.
- The RESTORE command is executed and the input volume is created by a physical dump operation.

1.23.3.1 When to use physical processing

Use physical processing when:

- Backing up system volumes that you might want to restore with a stand-alone DFSMSdss restore operation.

  Stand-alone DFSMSdss restore supports only physical dump tapes.

- Performance is an issue.

  Generally, the fastest way—measured by elapsed time—to copy or to dump an entire volume is with a physical full-volume command. This is primarily because minimal catalog searching is necessary for physical processing.

- Substituting one physical volume for another or recovering an entire volume.

  With a COPY or RESTORE (full volume or track) command, the volume serial number of the input DASD volume can be copied to the output DASD volume.

- Dealing with I/O errors. Physical processing provides the capability to copy, dump, and restore a specific track or range of tracks.

- Dumping or copying between volumes of the same device type but different capacity.

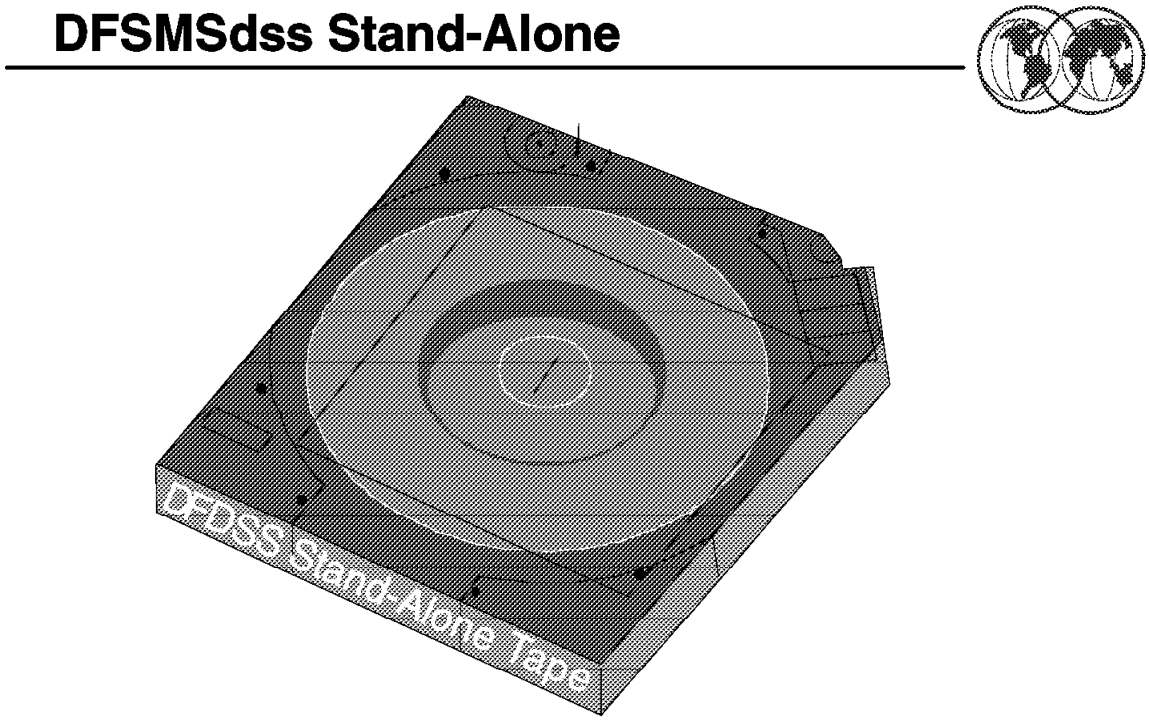


Figure 80. Stand-alone services

1.23.4 DFSMSdss stand-alone services

DFSMS/MVS Version 1 Release 4 provided a new stand-alone services function, that is intended for the storage administrator, the system programmer, and anyone who runs the stand-alone services program. This, along with related information in *OS/390 MVS System Messages, Volume 1 (ABA-ASA)*, GC28-1784, supports a new stand-alone services program.

stand-alone services can perform either a full-volume or a tracks restore from dump tapes produced by DFSMSdss or DFDSS, and offers the following benefits when compared to the previous DFSMSdss stand-alone functions:

- Provides user-friendly commands to replace the previous control statements
- Supports IBM 3494 and 3495 Tape Libraries, and 3590 Tape Subsystems
- Supports IPLing from a DASD volume, in addition to tape and card readers
- Allows you to predefine the operator console to be used during stand-alone services processing

For detailed information about the stand-alone service, and other DFSMSdss information, refer to *DFSMS/MVS DFSMSdss Storage Administration Reference*, SC26-4929, and *DFSMS/MVS DFSMSdss Storage Administration Guide*, SC26-4930.

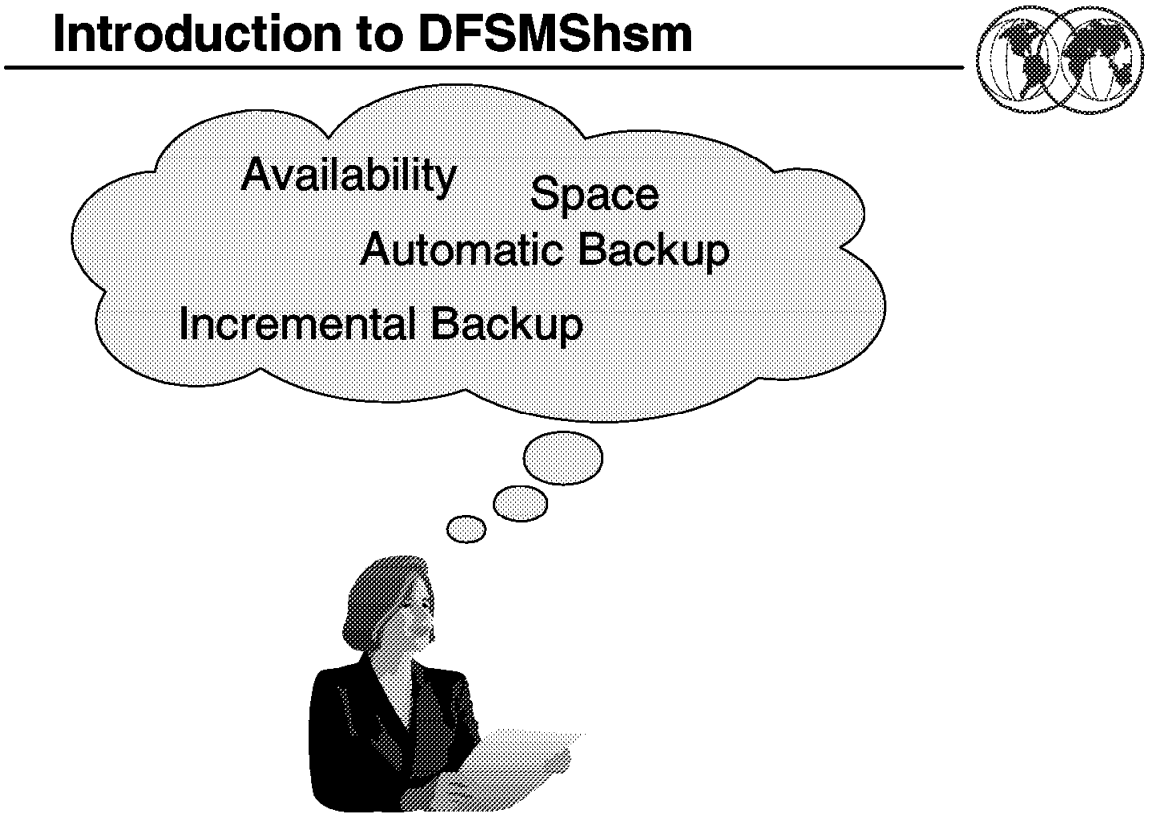


Figure 81. Introduction to DFSMShsm

1.24 DFSMShsm

DFSMShsm is a licensed program that automatically performs *space management* and *availability management* in a storage device hierarchy. Availability management is used to make data available by automatically copying new and changed data set to backup volumes. Space management is used to manage DASD space by enabling inactive data sets to be moved off fast-access storage devices thus creating free space or new allocations. DFSMShsm also provides for other supporting functions that are essential to your installation's environment.

If you need further information about DFSMShsm, refer to *DFSMShsm Storage Administration Guide*, SH21-1076, and *DFSMShsm Storage Administration Reference*, SH21-1075.

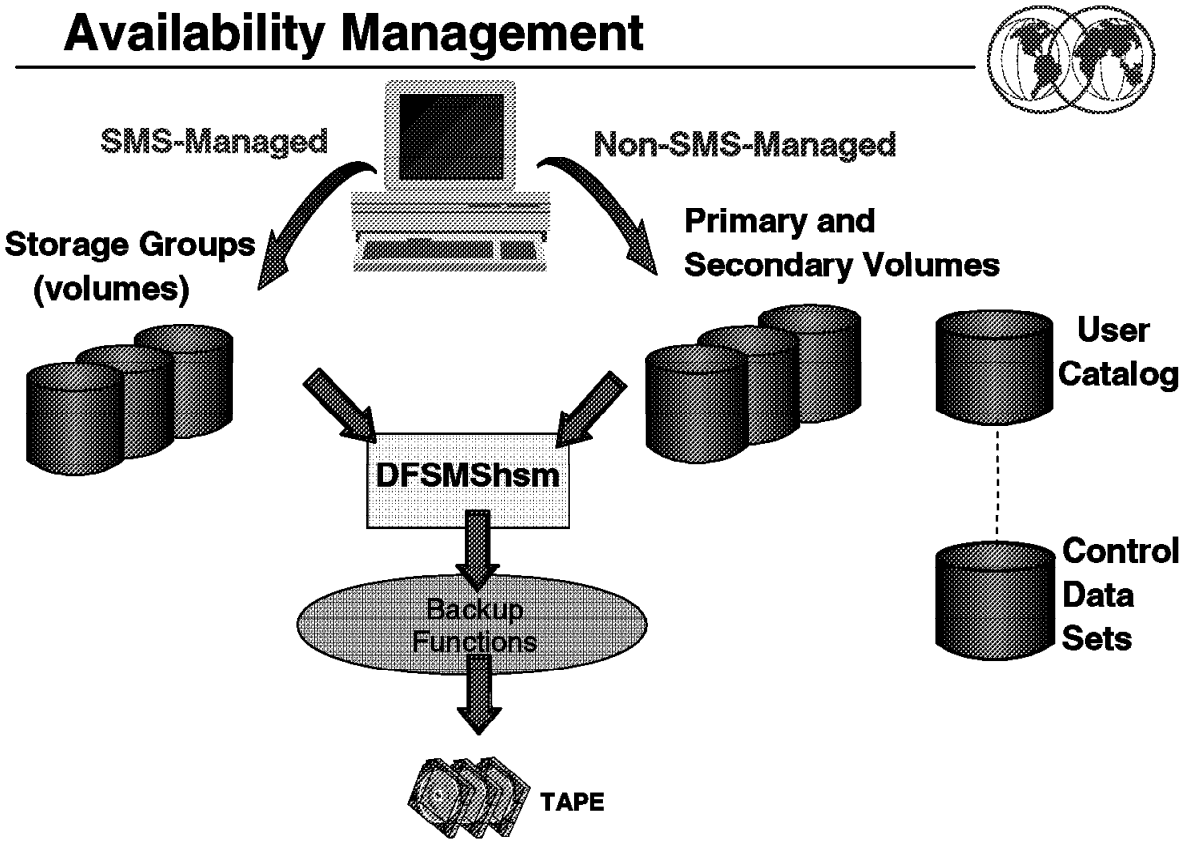


Figure 82. Availability management

1.24.1 Availability management

Availability management ensures that a recent copy of your DASD data set exists. The purpose of availability management is to ensure that lost or damaged data sets can be retrieved at the most current possible level. To do this, availability management automatically and periodically performs functions that:

1. Copy all the data sets on DASD volumes to tape volumes
2. Copy the changed data sets on DASD volumes (incremental backup) either to other DASD volumes or to tape volumes

DFSMShsm minimizes the space occupied by the data sets on the backup volume.

Availability management functions are:

- Automatic physical full-volume dump
- Automatic incremental backup
- Automatic control data set backup
- Command dump and backup
- Command recovery
- Expiration of backup versions
- Disaster backup

- Aggregate backup and recovery (ABARS)

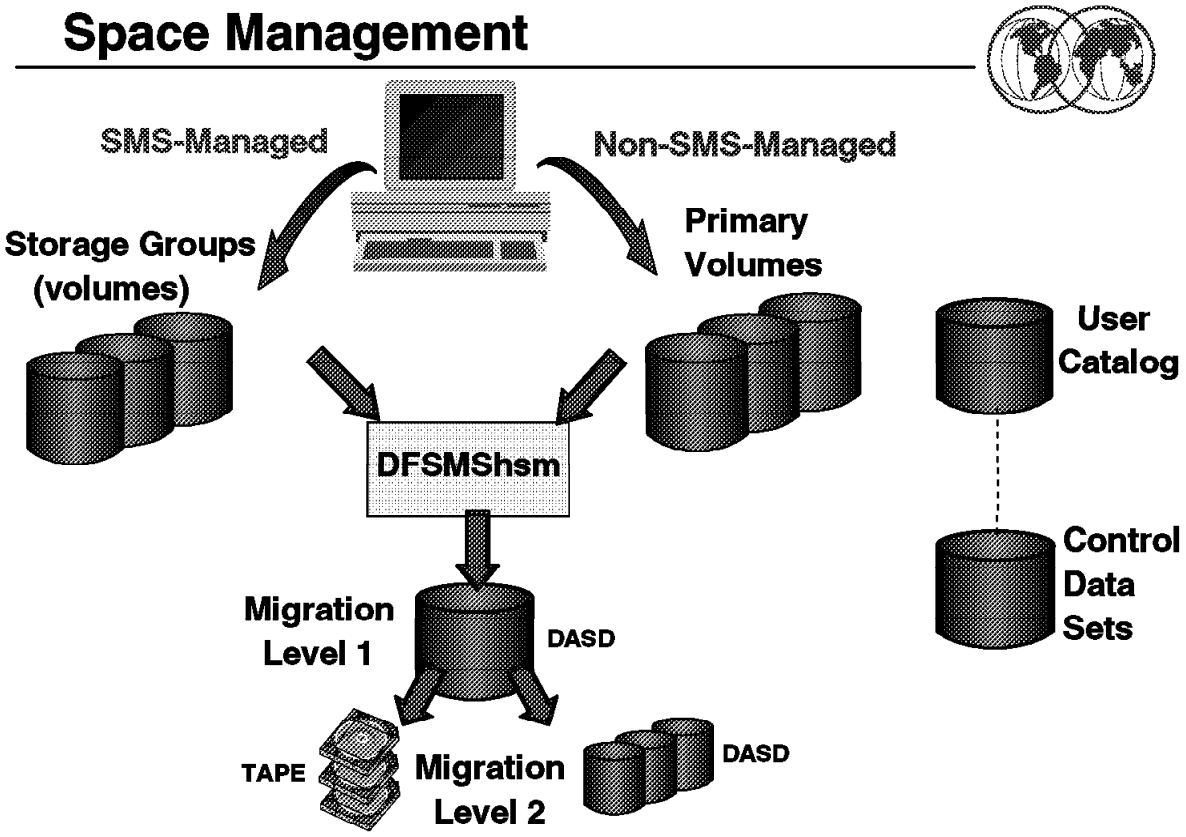


Figure 83. Space management

1.24.2 Space management

Space management is the function of DFSMShsm that allows you to keep DASD space available for users in order to meet the service-level objectives for your system. The purpose of space management is to manage your DASD storage efficiently. To do this, space management automatically and periodically performs functions that:

1. Move low-activity data sets from user-accessible volumes to DFSMShsm volumes.
2. Reduce the space occupied by data on both the user-accessible volumes and the DFSMShsm volumes.

The DFSMShsm space management functions are:

- Automatic primary space management of DFSMShsm-managed volumes, which includes:
  - Deletion of temporary data sets
  - Deletion of expired data sets
  - Release of unused, over-allocated space
  - Migration to DFSMShsm-owned migration level-1 (ML1) volumes (compressed)
- Automatic secondary space management of DFSMShsm-owned volumes, which includes:
  - ML1 cleanup, including deletion of expired migrated data sets and some migration control data set (MCDS) records
  - Moving migration copies from migration level 1 (ML1) to migration level 2 (ML2) volumes

- Automatic interval migration, initiated when a DFSMShsm-managed volume exceeds a specified threshold
- Automatic recall of user data sets back to DASD volumes, when referenced by the application
- Space management by command
- Space-saving functions, which include:
  - Data compaction/compression. Compaction provides space savings that are due to less gaps and less control data. Compression provides a more short way for storing your data.
  - Partitioned data set (PDS) free space compression.
  - Small data set packing (SDSP) data set facility, allows small data sets be packaged in just one physical track.
  - Data set reblocking.

It is possible to have more than one OS/390 image sharing the same DFSMShsm policy. In this case one of the DFSMShsm images is the primary host and the others are secondary. The primary HSM host is identified by ′HOST= in the HSM startup and is responsible for:

- Hourly space checks
- During autobackup: CDS BUP, BUP of ML1 data sets to tape
- During autodump: expiration of dump copies and deletion of excess dump VTOC copy data sets
- During SSM: cleanup of MCDS, migration volumes, and L1 to L2 migration

If you are running your OS/390 HSM images in sysplex (parallel or basic), you can use *secondary host promotion* to allow a secondary image to assume the primary image′s tasks if the primary host fails. Secondary host promotion uses XCF status monitoring to execute the promotion. To indicate a system as a candidate, issue:

- SETSYS PRIMARYHOST(YES) and
- SSM(YES)

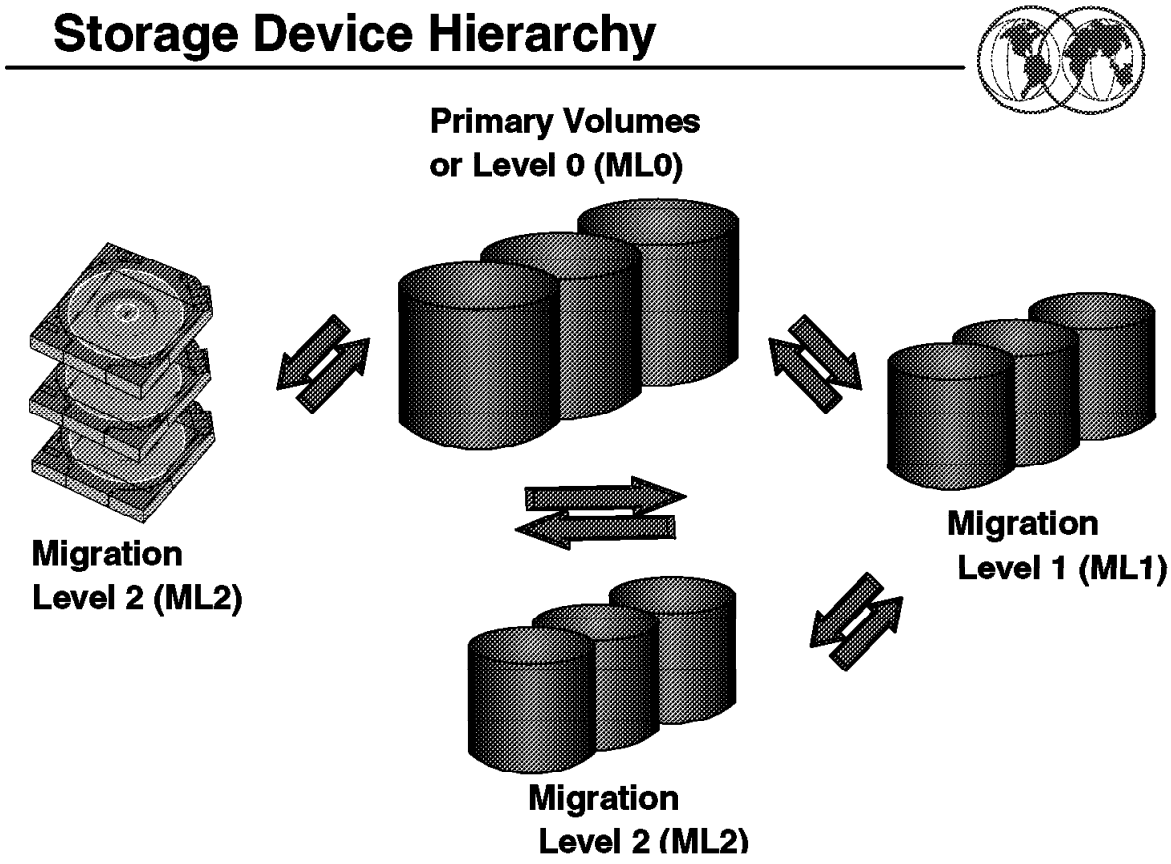


Figure 84. Storage device hierarchy

1.24.3 Storage device hierarchy

A storage device hierarchy consists of a group of storage devices that have different costs for storing data, different amounts of data stored, and different speeds of accessing the data.

DFSMShsm uses the following three-level storage device hierarchy for space management:

- Level 0: Are DFSMShsm-managed storage devices at the highest level of the hierarchy; these devices contain data directly accessible to your application
- Level 1 and Level 2: Storage devices at the lower levels of the hierarchy, level 1 and level 2, contain data that DFSMShsm has compressed and optionally compacted into a format that you cannot use. Devices at this level provide lower-cost-per-byte storage and usually slower response time. Usually L1 is in a cheaper DASD (or a same cost, but with the gain of compression) and L2 is on tape.

  If you have RVA DASD you may skip level one (ML1) migration because the data in L0 is already compacted/compressed.

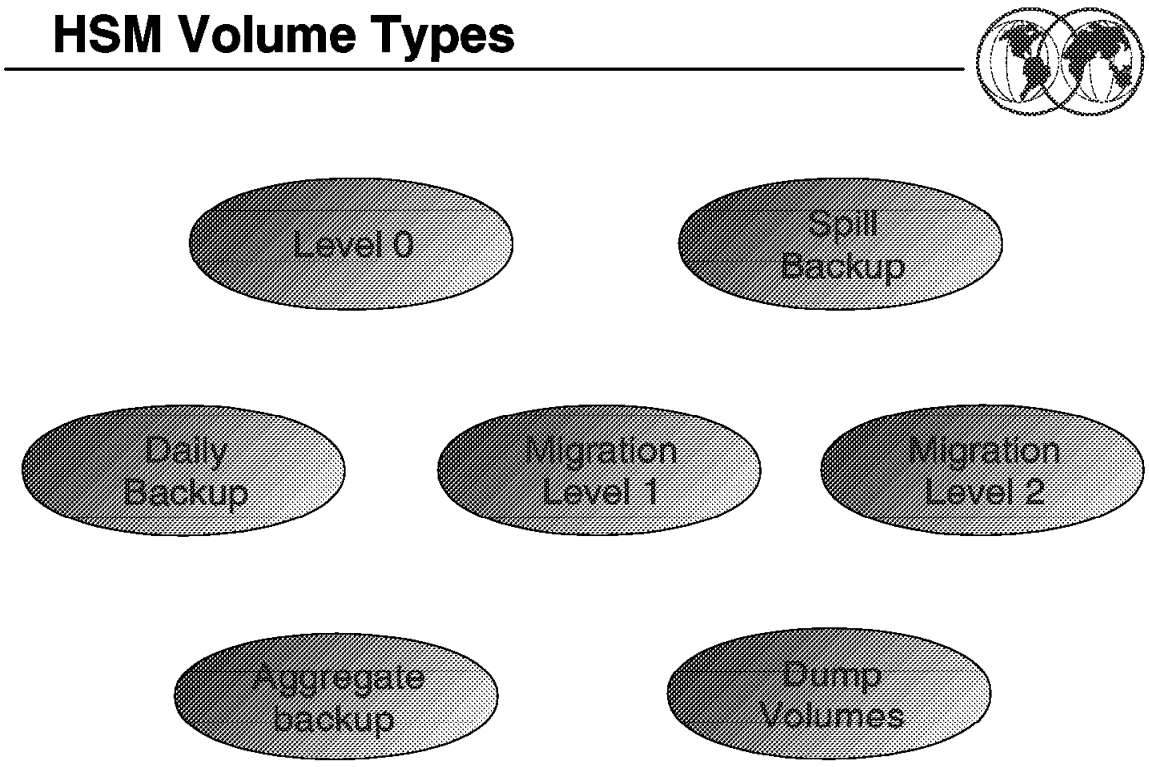


Figure 85. HSM volume types

1.24.4 HSM volume types

Backing up an individual cataloged data set is performed in the same way as for SMS-managed data sets. However, to back up individual uncataloged data sets, issue the following command:

```
BACKDS dsname UNIT(unittype) VOLUME(volser)

HBACKDS dsname UNIT(unittype) VOLUME(volser)
```

The HBACKDS form of the command can be used by either non-DFSMShsm-authorized or DFSMShsm-authorized users. The BACKDS form of the command can be used only by DFSMShsm-authorized users. The UNIT and VOLUME parameters are required because DFSMShsm cannot locate an uncataloged data set without being told where it is.

DFSMShsm supports the following volume types:

- Level 0 (L0) volumes contain data sets that are directly accessible to you and the jobs you run. DFSMShsm-managed volumes are those L0 volumes that are managed by the DFSMShsm automatic functions. These volumes must be mounted and online when you refer to them with DFSMShsm commands.
- Migration level 1 (ML1) volumes are DFSMShsm-supported DASD on which DFSMShsm maintains your data in DFSMShsm format. These volumes are normally permanently mounted and online. They can be:
  - Volumes containing data sets that DFSMShsm migrated from L0 volumes.

  - Volumes containing backup versions created from a DFSMShsm BACKDS or HBACKDS command. Backup processing requires ML1 volumes to store incremental back up and dump VTOC copy data sets and as intermediate storage for data sets that are backed up by data set command backup.

- Migration level 2 (ML2) are DFSMShsm-supported tape, or DASD, on which DFSMShsm maintains your data in DFSMShsm format. These volumes are normally not mounted or online. They contain data sets migrated from ML1 volumes or L0 volumes.
- Daily backup volumes are DFSMShsm-supported tape, or DASD, on which DFSMShsm maintains your data in DFSMShsm format. These volumes are normally not mounted or online. They contain the most current backup versions of data sets copied from L0 volumes. These volumes may also contain earlier backup versions of these data sets.
- Spill backup volumes are DFSMShsm-supported tape, or DASD, on which DFSMShsm maintains your data sets in DFSMShsm format. These volumes are normally not mounted or online. They contain earlier backup versions of data sets, which were moved from DASD backup volumes.
- Dump volumes are DFSMShsm-supported tape. They contain image copies of volumes, which are produced by the full volume dump function of DFSMSdss (write a copy of the entire allocated space of that volume), which is invoked by DFSMShsm.
- Aggregate backup volumes are DFSMShsm-supported tape. These volumes are normally not mounted or online. They contain copies of the data sets of a user-defined group of data sets, along with control information for those data sets. These data sets and their control information are stored as a group so that they can be recovered (if necessary) as an entity by an aggregate recovery process (ABARS).

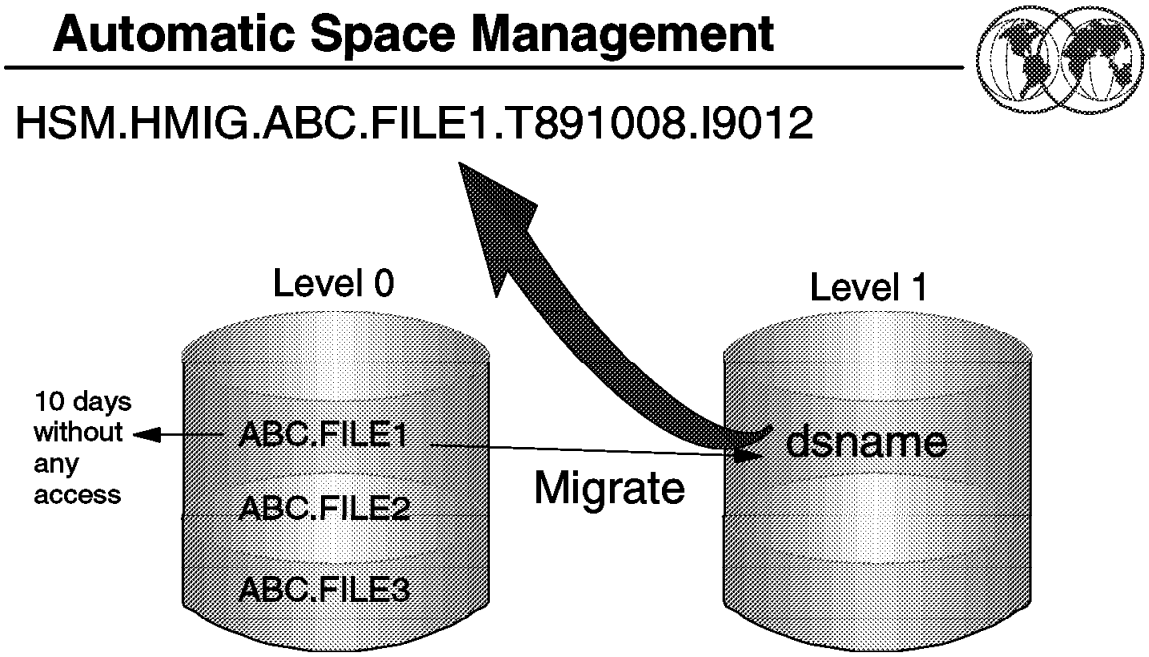


Figure 86. Automatic space management

1.24.5 Automatic space management

Automatic space management prepares the computing system for the addition of new data by freeing space on the DFSMShsm-managed volumes (L0) and DFSMShsm-owned volumes (ML1).

The functions associated with automatic space management can be divided into two groups:

- Automatic volume space management
  - Primary

    It is invoked timely in a daily basis. It cleans L0 volumes by deleting expired and temporary data sets and releasing allocated and not used space. If after that, the free space is still below a threshold, then it moves data sets (under control of the management class) from L0 to ML1 or ML2 volumes.
  - Interval migration

    It is executed each hour throughout the day, as needed for all storage groups. In interval migration, DFSMShsm performs space check on each DFSMShsm volume being managed. A volume is considered eligible for interval migration based on the AUTOMIGRATE and THRESHOLD settings.
- Automatic secondary space management

  It deletes expired data sets from ML1/ML2, then moves data sets (under control of the management class) from ML1 to ML2 volumes. It should complete before automatic primary space management so that the ML1 volumes will not run out of space.

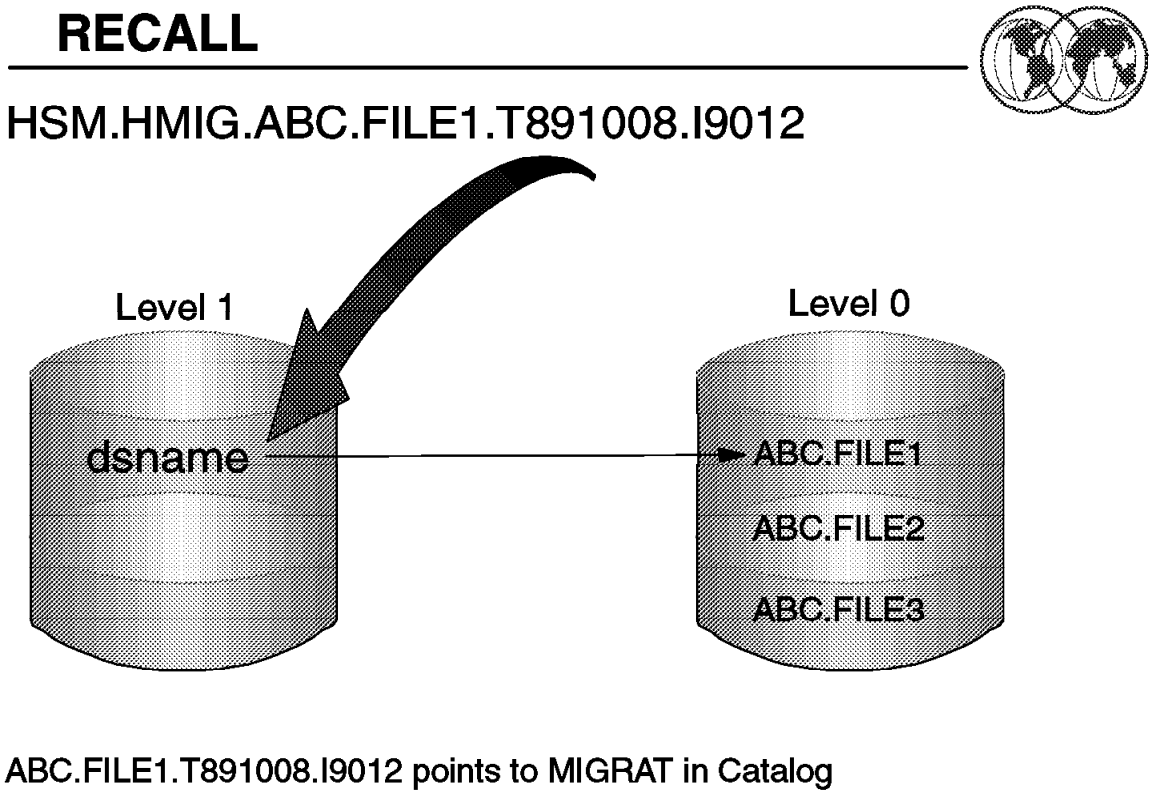


Figure 87. Recall

1.24.6 Recall

Recall returns a migrated data set to a user L0 volume. The recall is transparent and the application does not need to know that it happened and where the migrated data set resides. To provide applications with quick access to their migrated data sets, DFSMShsm allows up to 15 concurrent recall tasks. RMF monitor III shows delays caused by the recall operation.

The MVS allocation routine discovers that the data set is migrated when accessing the catalog, it finds instead of the volser, the word MIGRAT.

Automatic recall returns your migrated data set to a DFSMShsm-managed volume when you refer to it. The catalog is updated accordingly.

Command recall returns your migrated data set to a user volume when you enter the HRECALL command through an ISMF panel or by directly keying in the command.

For both automatic and command recall, DFSMShsm working with SMS invokes the automatic class selection (ACS) routines. Data sets that were not SMS-managed at the time they were migrated may be recalled as SMS-managed data sets. The ACS routines determine whether the data sets should be recalled as SMS-managed, and if so, the routines select the classes and storage groups in which the data sets will reside. The system chooses the appropriate volume for the data sets.

DFSMShsm working without SMS returns a migrated data set to a DFSMShsm-managed non-SMS level 0 volume with the most free space.

The recall operation can also be done explicitly by a DFSMShsm command.

Introduction to DFSMSrmm

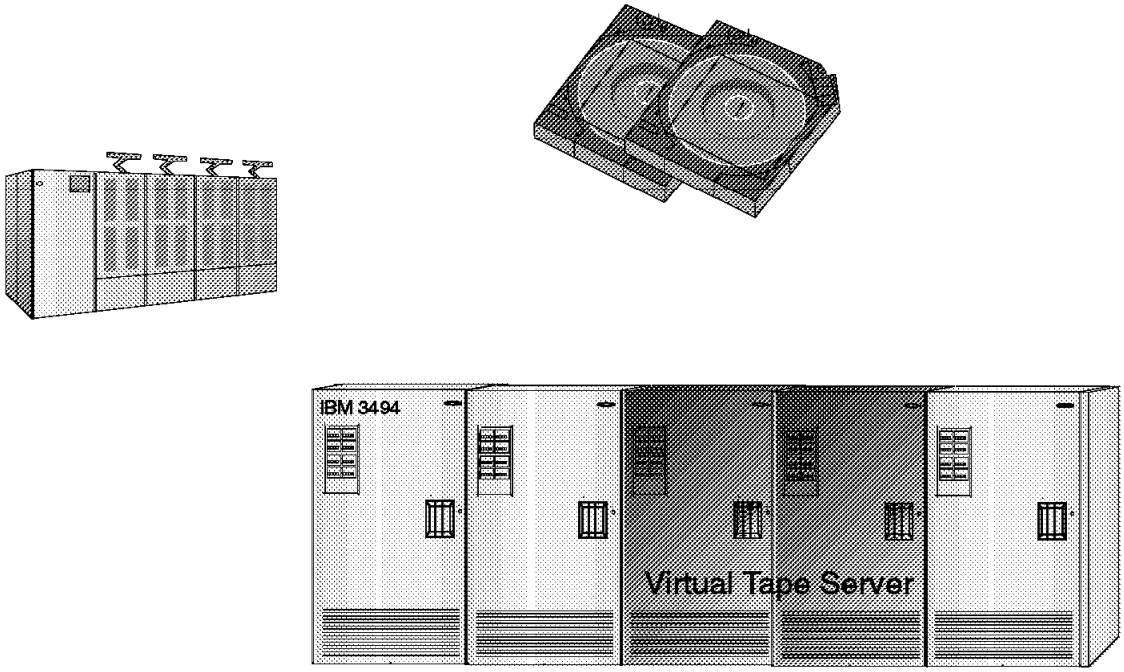


Figure 88. Introduction to DFSMSrmm

1.25 Removable media manager (DFSMSrmm)

In your enterprise, you store and manage your removable media in several types of media libraries. For example, in addition to your traditional tape library, a room with tapes, shelves, and drives, you might have several automated and manual tape libraries. You probably also have both on-site libraries and off-site storage locations, also known as vaults or stores.

With the DFSMSrmm functional component of DFSMS/MVS, you can manage your removable media as one enterprise-wide library (single image) across systems. Due to the need of global control information these systems must have accessibility to some shared DASD volumes. DFSMSrmm manages your installation′s tape volumes and the data sets on those volumes. DFSMSrmm also manages the shelves where volumes reside in all locations except in automated tape library dataservers.

DFSMSrmm manages all tape media, such as cartridge system tapes and 3420 reels, as well as other removable media you define to it. For example, DFSMSrmm can record the shelf location for optical disks and track their vital record status; however it does not manage the objects on optical disks.

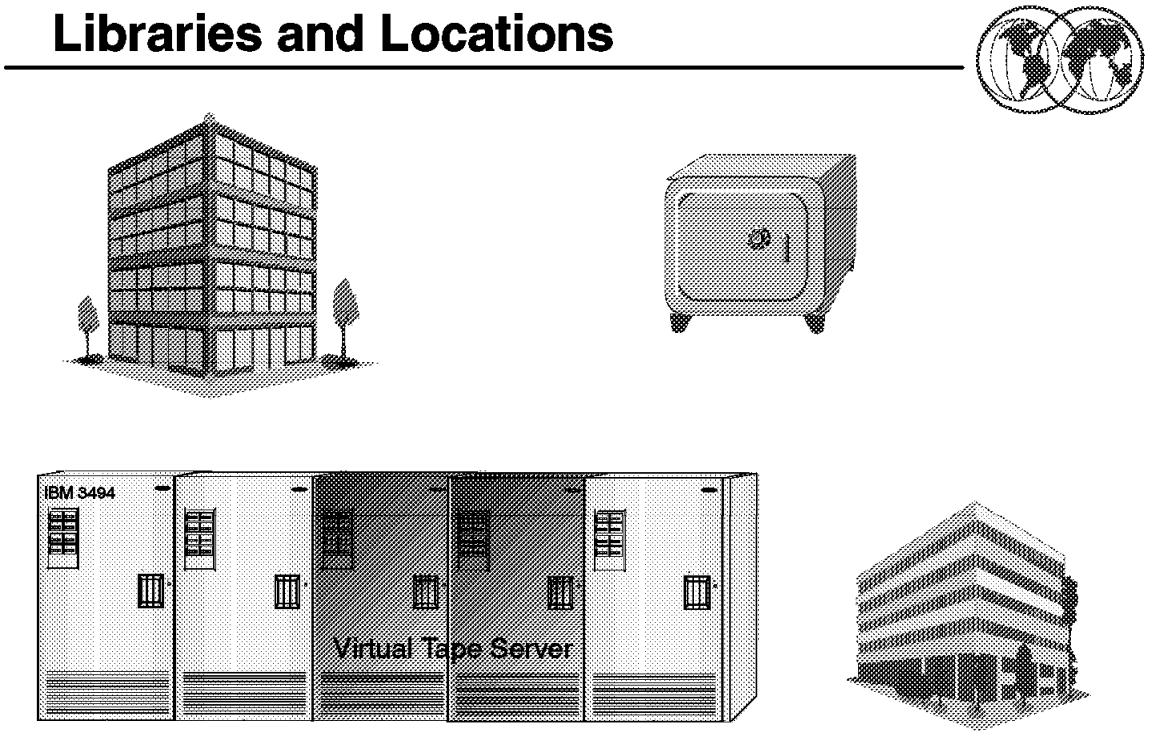


Figure 89. Libraries and locations

1.25.1 Libraries and locations

You decide where to store your removable media based on how often the media is accessed and for what purpose it is retained. For example, you might keep volumes that are frequently accessed in an automated tape library dataserver, and you probably use at least one storage location to retain volumes for disaster recovery and audit purposes. You might also have locations where volumes are sent for further processing such as other data centers within your company or your customers and vendors.

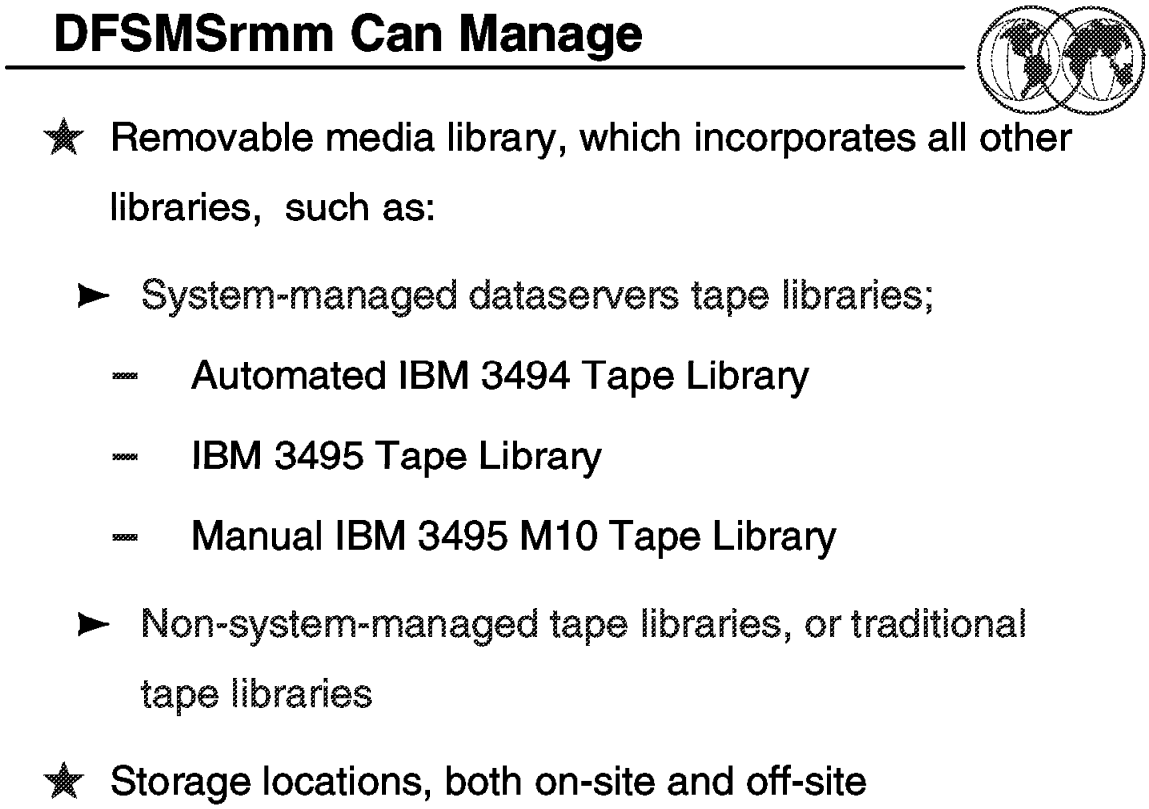


Figure 90. DFSMSrmm can manage

1.26 What DFSMSrmm can manage

DFSMSrmm can manage the following libraries and storage locations:

1.26.1 Removable media library

A removable media library contains all the tape and optical volumes that are available for immediate use, including the shelves where they reside. A removable media library usually includes other libraries: system-managed libraries such as automated or manual tape library dataservers; and non-system-managed libraries, containing the volumes, shelves, and drives not in an automated or a manual tape library dataserver.

In the removable media library, you store your volumes in shelves, where each volume occupies a single shelf location. This shelf location is referred to as a rack number in the DFSMSrmm TSO subcommands and ISPF dialog. A rack number matches the volume′s external label. DFSMSrmm uses the external volume serial number to assign a rack number when adding a volume, unless you specify otherwise. The format of the volume serial you define to DFSMSrmm must be one to six alphanumeric characters. The rack number must be six alphanumeric or national characters.

1.26.1.1 System-managed tape library

A system-managed tape library is a collection of tape volumes and tape devices defined in the tape configuration database. The tape configuration database is an integrated catalog facility user catalog marked as a volume catalog (VOLCAT) containing tape volumes and tape library records. A system-managed tape library can be either automated or manual:

- An *automated tape library dataserver* is a device consisting of robotic components, cartridge storage areas (or shelves), tape subsystems, and controlling hardware and software, together with the set of tape volumes that reside in the library and can be mounted on the library tape drives. The IBM automated tape libraries are the automated IBM 3494 and IBM 3495 Library Dataservers.
- A *manual tape library dataserver* is a set of tape drives and the set of system-managed volumes the operator can mount on those drives. The IBM manual tape library is the manual IBM 3495 Tape Library Dataserver, which supports 3490 and 3490E Magnetic Tape Subsystems.

1.26.1.2 Non-system-managed tape library

A non-system-managed tape library is all the volumes, shelves, and drives not in an automated tape library dataserver or manual tape library dataserver. You might know this library as the traditional *tape library*. DFSMSrmm provides complete tape management functions for the volumes and shelves in this traditional tape library. Volumes in a non-system-managed library are defined by DFSMSrmm as being shelf resident.

All tape media and drives supported by OS/390 are supported in this environment. Using DFSMSrmm, you can fully manage all types of tapes in a non-system-managed tape library, including 3420 reels, 3480, and 3590 cartridge system tapes.

1.26.2 Storage location

Storage locations are not part of the removable media library because the volumes in storage locations are not generally available for immediate use. A storage location is comprised of shelf locations that you define to DFSMSrmm. A shelf location in a storage location is identified by a bin number. Storage locations are typically used to store removable media that are kept for disaster recovery or vital records.

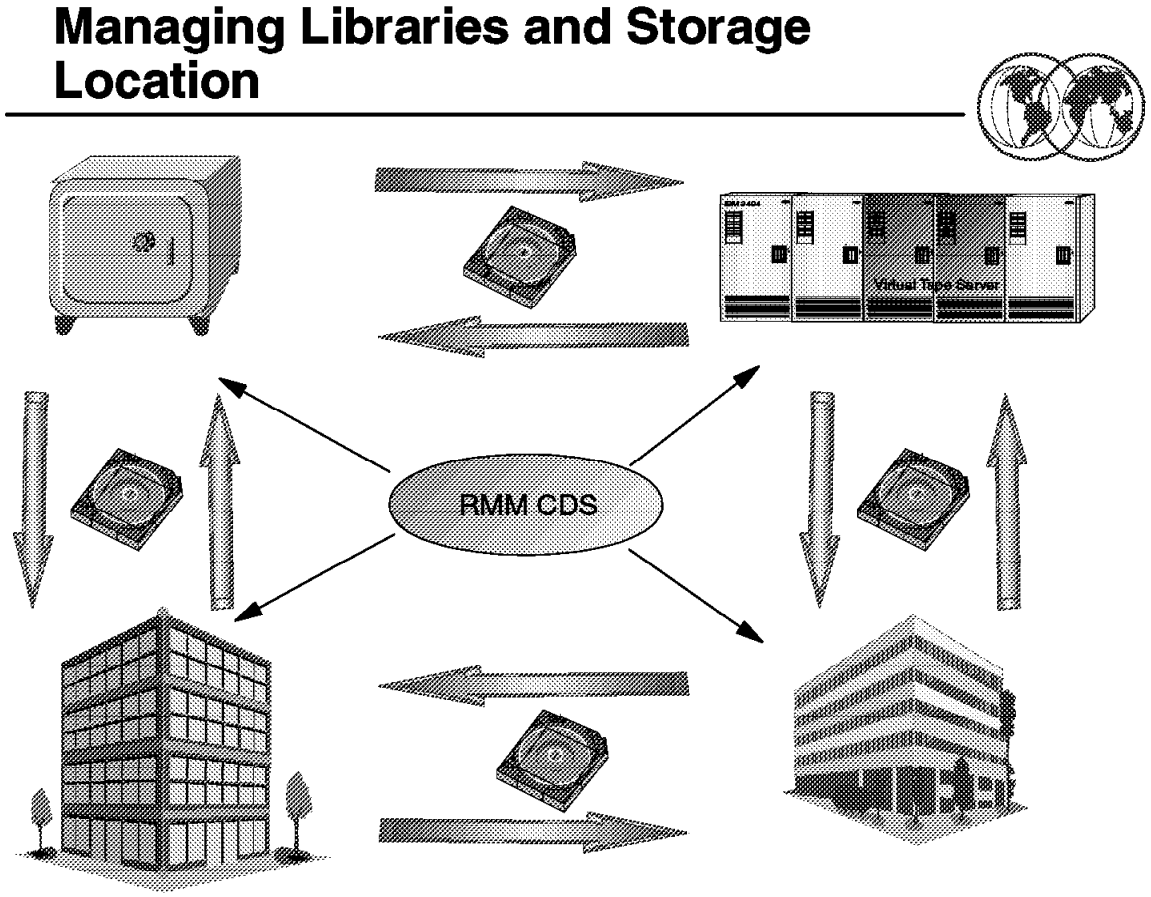


Figure 91. Managing libraries and storage locations

1.26.3 Managing libraries and storage locations

DFSMSrmm records the complete inventory of the removable media library and storage locations in the DFSMSrmm control data set—a VSAM key-sequenced data set. In the control data set, DFSMSrmm records all changes made to the inventory, such as adding or deleting volumes, and also keeps track of all movement between libraries and storage locations.

DFSMSrmm manages the movement of volumes among all library types and storage locations. This lets you control where a volume, and hence a data set, resides and how long it is retained.

DFSMSrmm helps you manage the movement of your volumes and retention of your data over their full life, from initial use to the time they are retired from service. Among the functions DFSMSrmm performs for you are:

- Automatically initializing and erasing volumes
- Recording information about volumes and data sets as they are used
- Expiration processing
- Identifying volumes with high error levels that require replacement

To make full use of all of the DFSMSrmm functions, you specify installation setup options and define retention and movement policies.

For more information about DFSMSrmm, refer to *DFSMS/MVS DFSMSrmm Guide and Reference*, SC26-4931, and *DFSMS/MVS DFSMSrmm Implementation and Customization Guide*, SC26-4932.

Chapter 2. Storage management

This chapter is intended to help you learn about and evaluate DFSMS/MVS and is an overview of DFSMS/MVS and its functional components.

DFSMS/MVS is built upon the functions formerly provided by MVS/DFP, Data Facility Data Set Services (DFDSS), and the Data Facility Hierarchical Storage Manager (DFHSM). DFSMS/MVS eliminates the need to order and install each product individually, making the installation task much easier.

As your business expands, so do your needs for storage to hold your applications and data, and the costs of managing that storage. Storage costs include more than the price of the hardware, with the highest cost being the people needed to perform storage management tasks. If your business requires transaction systems, the batch window can also be a high cost. Additionally, you must pay for people to install, monitor, and operate your storage hardware devices, for electrical power to keep each piece of storage hardware cool and running, and for floor space to house them. Removable media, such as optical and tape storage, cost less per gigabyte (GB) than online storage, but require additional time and resources to locate, retrieve, and mount.

To allow your business to grow efficiently and profitably, you want to find ways to control the growth of your information systems and use your current storage more effectively.

The DFSMS/MVS software product, together with IBM hardware products, and your installation-specific requirements for data and resource management comprise the key to system-managed storage in an MVS environment. The components of DFSMS/MVS automate and centralize storage management, based on policies your installation defines for availability, performance, space, and security. The Interactive Storage Management Facility (ISMF) provides the user interface for defining and maintaining these policies, which the Storage Management Subsystem (SMS) governs for the system.

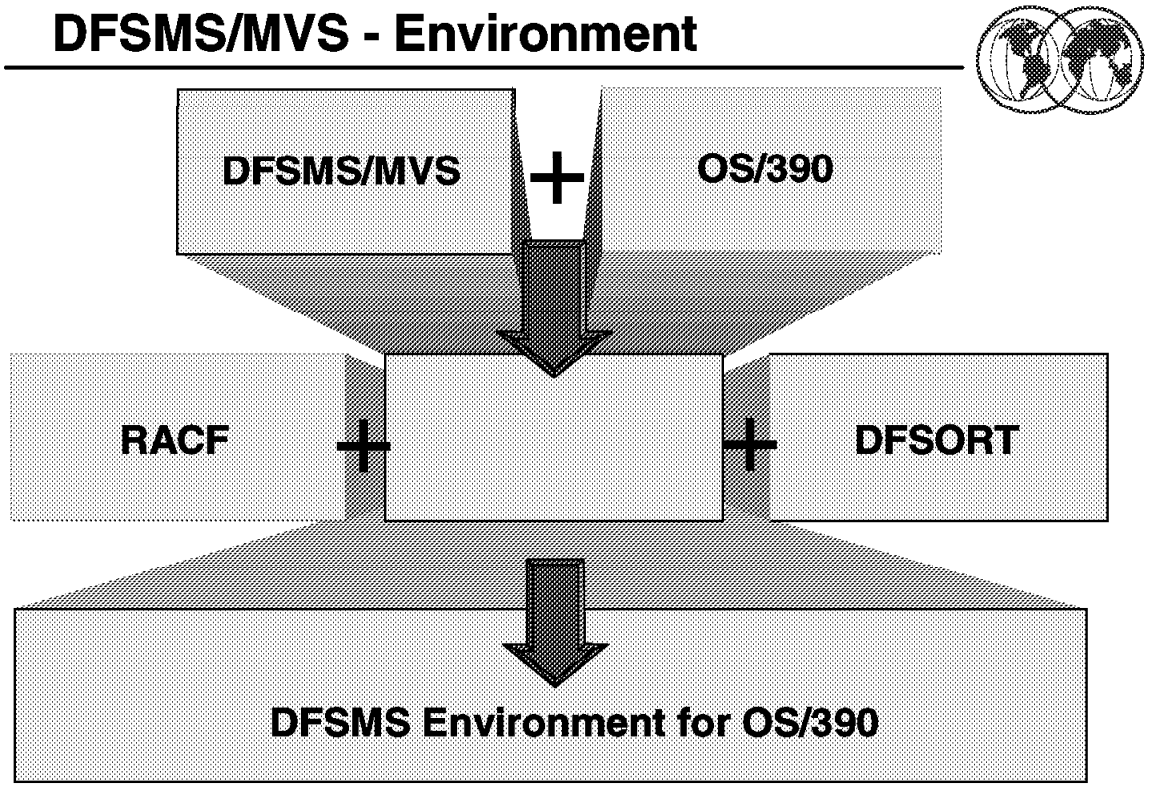


Figure 92. DFSMS/MVS environment

2.1 DFSMS/MVS environment

As your business expands, so do your needs for storage to hold your applications and data, and the costs of managing that storage. Storage costs include more than the price of the hardware, with the highest cost being the people needed to perform storage management tasks. If your business requires transaction systems, the batch window can also be a high cost. Additionally, you must pay for people to install, monitor, and operate your storage hardware devices, for electrical power to keep each piece of storage hardware cool and running, and for floor space to house them. Removable media, such as optical and tape storage, cost less per gigabyte (GB) than online storage, but require additional time and resources to locate, retrieve, and mount.

To allow your business to grow efficiently and profitably, you want to find ways to control the growth of your information systems and use your current storage more effectively.

The DFSMS/MVS software product, together with IBM hardware products, and your installation-specific requirements for data and resource management comprise the key to system-managed storage in an MVS environment. The components of DFSMS/MVS automate and centralize storage management, based on policies your installation defines for availability, performance, space, and security. The Interactive Storage Management Facility (ISMF) provides the user interface for defining and maintaining these policies, which the Storage Management Subsystem (SMS) governs for the system.

This unit gives a brief overview of DFSMS/MVS, its components, available configurations, and functions.

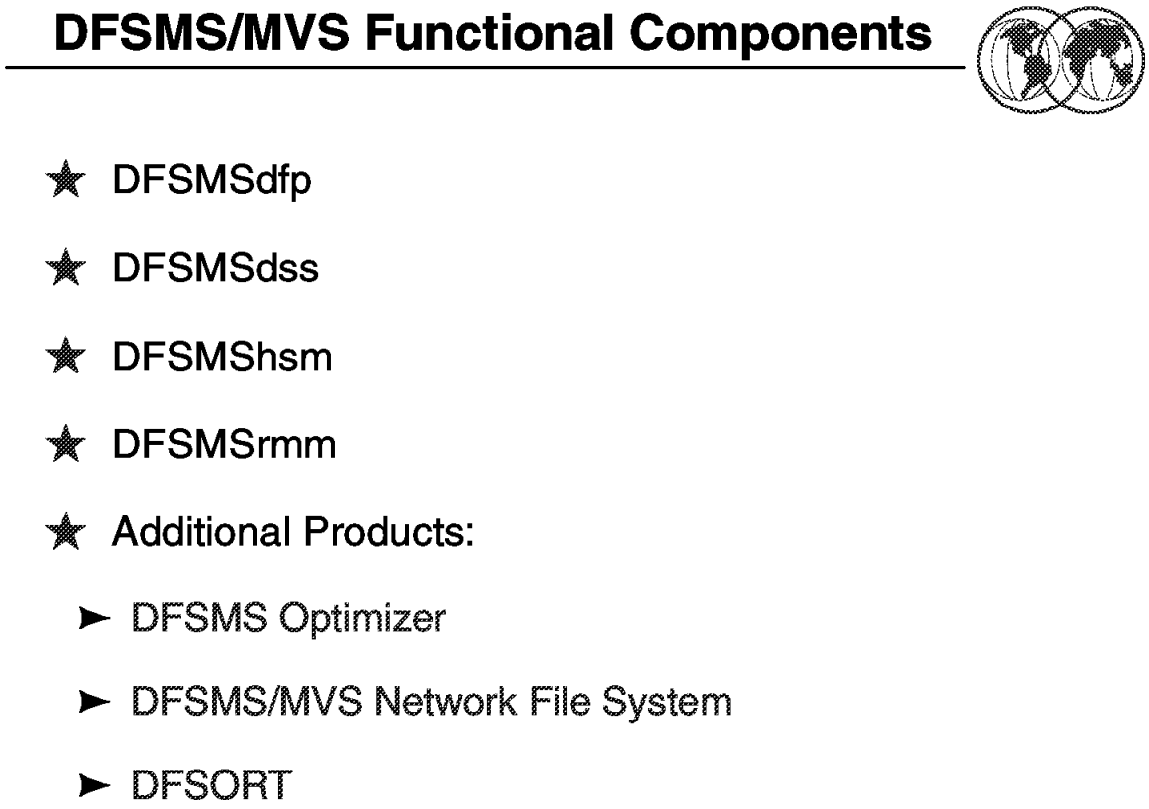


Figure 93. DFSMS/MVS functional components

2.1.1 The DFSMS/MVS functional components

This visual lists the components of DFSMS/MVS.

Do not confuse DFSMS/MVS with DFSMS (also called SMS) environment. DFSMS/MVS is a set of products (where one of these, DSFMSdfp, is mandatory to run OS/390). DFSMS environment is a set of constructs, user interfaces, and routines (using the DFSMS/MVS products), which allows your installation to better manage your storage system. All the core logic of DFSMS is located in DFSMSdfp, such as the ACS routines, ISMF code, and constructs. DFSMShsm and DFSMSdss are involved in the management class construct.

DFSMS/MVS and MVS comprise the base OS/390 operating system where DFSMS/MVS performs the essential data, storage, program, and device management functions of the system.

DFSMS/MVS is the central component of both system-managed and non-system-managed storage environments. MVS supports both 24-bit and 31-bit addressing used by components of DFSMS/MVS. Many DFSMS/MVS components have modules or data in extended virtual storage above 16 MB, leaving more space below the 16 MB line for user applications.

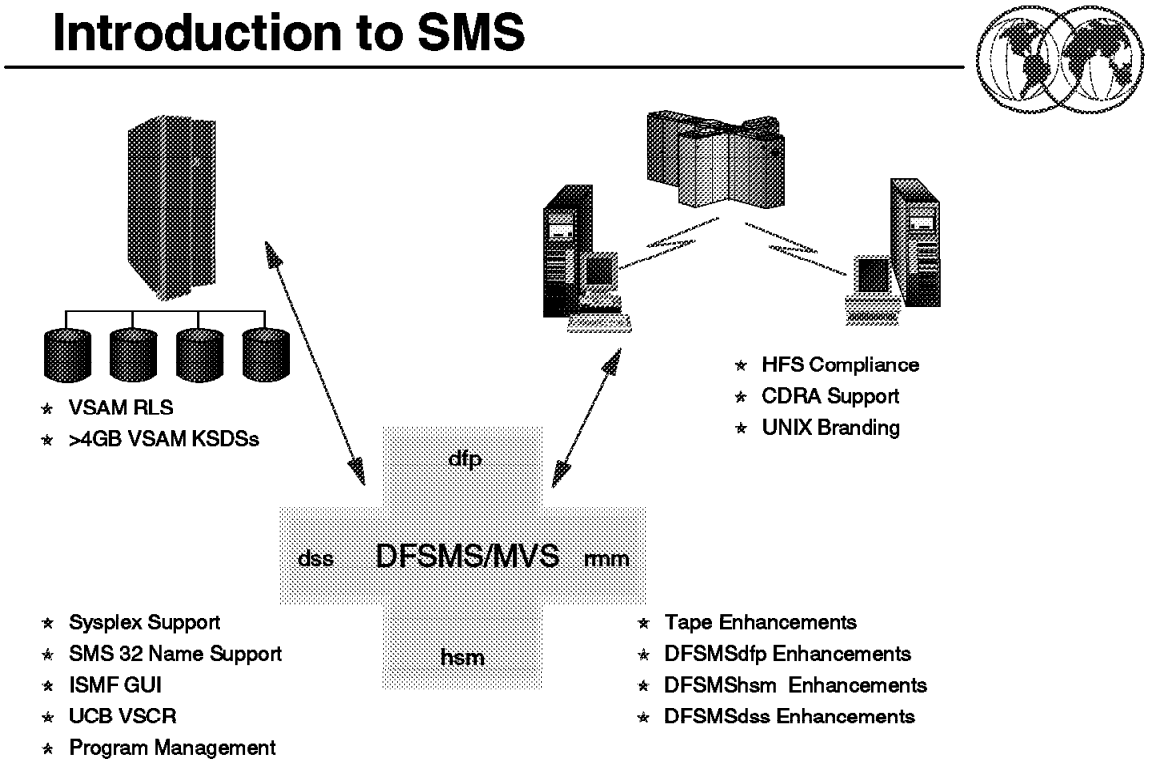


Figure 94. System-managed storage environment

2.2 Introduction to system-managed storage (SMS)

The DFSMS environment consists of a set of IBM hardware and software products that together provide a system-managed storage solution for MVS installations. DFSMS/MVS is an integral part of this environment.

The components of DFSMS/MVS automate and centralize storage management based on installation-defined policies for availability, performance, space, and security. The Interactive Storage Management Facility (ISMF) provides the user interface for defining and maintaining these policies and the Storage Management Subsystem (SMS) governs these policies for the system.

In this environment, the Resource Access Control Facility (RACF) and Data Facility Sort (DFSORT) products complement the functions of the base operating system. RACF provides resource security functions, and DFSORT adds the capability for faster and more efficient sorting, merging, copying, reporting, and analyzing of business information.

This visual also shows some of the recent enhancements of the DFSMS/MVS set of products.

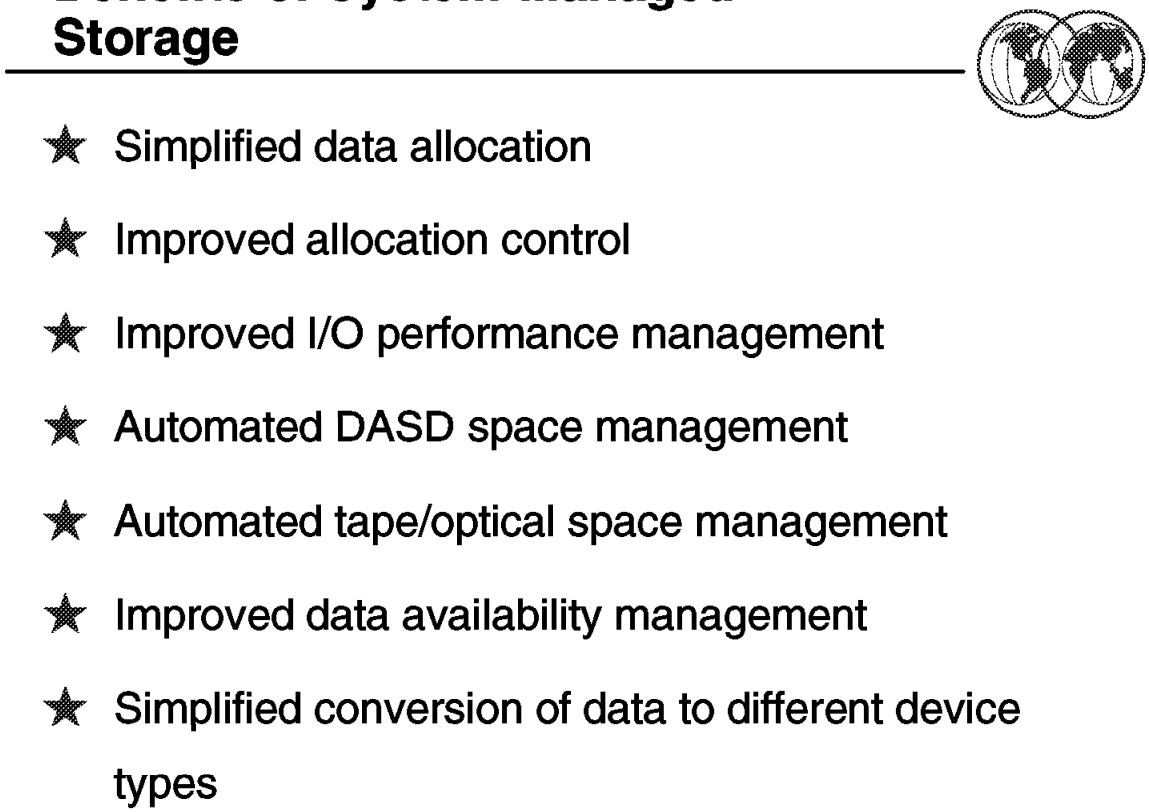


Figure 95. Benefits of system-managed storage

2.3 Benefits of system-managed storage

With the Storage Management Subsystem (SMS), you can define performance goals and data availability requirements, create model data definitions for typical data sets, and automate data backup. SMS can automatically assign, based on installation policy, those services and data definition attributes to data sets when they are created. IBM storage management-related products determine data placement, manage data backup, control space usage, and provide data security.

The goals of system-managed storage are to:

- Improve the use of the storage media; for example, by reducing out-of-space abends and providing a way to set a free-space requirement.
- Reduce the labor involved in storage management by centralizing control, automating tasks, and providing interactive controls for storage administrators.
- Reduce the user′s need to be concerned with the physical details, performance, space, and device management. Users can focus on using data instead of managing data.

The benefits, which may be integrated with the goals, of system-managed storage are:

Simplified data allocation: System-managed storage enables users to simplify their data allocations. For example, without using the Storage Management Subsystem, an OS/390 user would have to specify the unit and volume on which the system should allocate the data set. The user would also have to calculate the amount of space required for the data set in terms of tracks or cylinders. This means the

user has to know the track size of the device which will contain the data set. With system-managed storage, users can let the system select the specific unit and volume for the allocation. They can also specify size requirements in terms of megabytes or kilobytes. This means the user does not need to know anything about the physical characteristics of the devices in the installation.

Improved allocation control: System-managed storage enables you to set a requirement for free space across a set of direct access storage device (DASD) volumes. You can then provide adequate free space to avoid out-of-space abends. The system automatically places data on a volume containing adequate free space. In DFSMS/MVS 1.4 there are enhancements to avoid out-of-space by the relief of the SPACE requirements. You can also set a threshold for scratch tape volumes in tape libraries, to ensure enough cartridges are available in the tape library for scratch mounts.

Improved Input/Output (I/O) performance management: System-managed storage enables you to improve DASD I/O performance across the installation and at the same time reduce the need for manual tuning by defining performance goals for each class of data. You can use cache statistics recorded in System Management Facility (SMF) records to help evaluate performance. You can also improve sequential performance by using extended sequential data sets. The DFSMS environment makes the most effective use of the caching abilities of the IBM 3990 Model 3 and Model 6 Storage Controls, as well as other new models.

Automated DASD space management: System-managed storage enables you to automatically reclaim space which is allocated to old and unused data sets or objects. You can define policies that determine how long an unused data set or object will be allowed to reside on primary storage (storage devices used for your active data). You can have the system remove obsolete data by migrating the data to other DASD, tape, or optical volumes, or you can have the system delete the data. You can also release allocated but unused space which is assigned to new and active data sets.

Automated tape space management: System-managed storage enables you to fully use the capacity of your tape cartridges and to automate tape mounts. Using tape mount management techniques, DFSMShsm can fill tapes to their capacity. With 3490E tape devices, Enhanced Capacity Cartridge System Tape, 36-track recording mode, and the improved data recording capability, you can increase the amount of data that can be written on a single tape cartridge.

You can also use the IBM 3495 or 3494 Tape Library Dataserver to automatically mount tape volumes and manage the inventory in an automated tape library. If you do not have an automated tape library dataserver, you can still take advantage of system-managed tape by using manual tape libraries and the 3495 Model M10 Tape Library Dataserver.

Automated optical space management: System-managed storage enables you to fully use the capacity of your optical cartridges and to automate optical mounts. Using a 3995 Optical Library Dataserver, you can automatically mount optical volumes and manage the inventory in an automated optical library.

Improved data availability management: System-managed storage enables you to provide different backup requirements to data residing on the same DASD volume. Thus, you do not have to treat all data on a single volume the same way.

You can use DFSMShsm to automatically back up CICS/ESA and DATABASE 2 (DB2) databases, partitioned data sets extended (PDSEs), and physical sequential, partitioned, virtual storage access method (VSAM), hierarchical file system (HFS), and direct access data sets. You can also back up other types of data and use concurrent copy to maintain access to critical data sets while they are being backed up. Concurrent Copy, along with Backup-While-Open, has an added advantage that it avoids the invalidation of a backup of a CICS VSAM KSDS due to a control area or control interval split.

You can also create a logical grouping of data sets, so that the group is backed up at the same time to allow for recovery of the application defined by the group. This is done with the aggregate backup and recovery support (ABARS) provided by DFSMShsm.

Simplified conversion of data to different device types: System-managed storage enables you to move data to new volumes without requiring users to update their job control language (JCL). Because users in a DFSMS environment do not need to specify the unit and volume which contains their data, it does not matter to them if their data resides on a specific volume or device type. This allows you to easily replace old devices with new ones.

You can also use system-determined block sizes to automatically reblock physical sequential and partitioned data sets that can be reblocked.

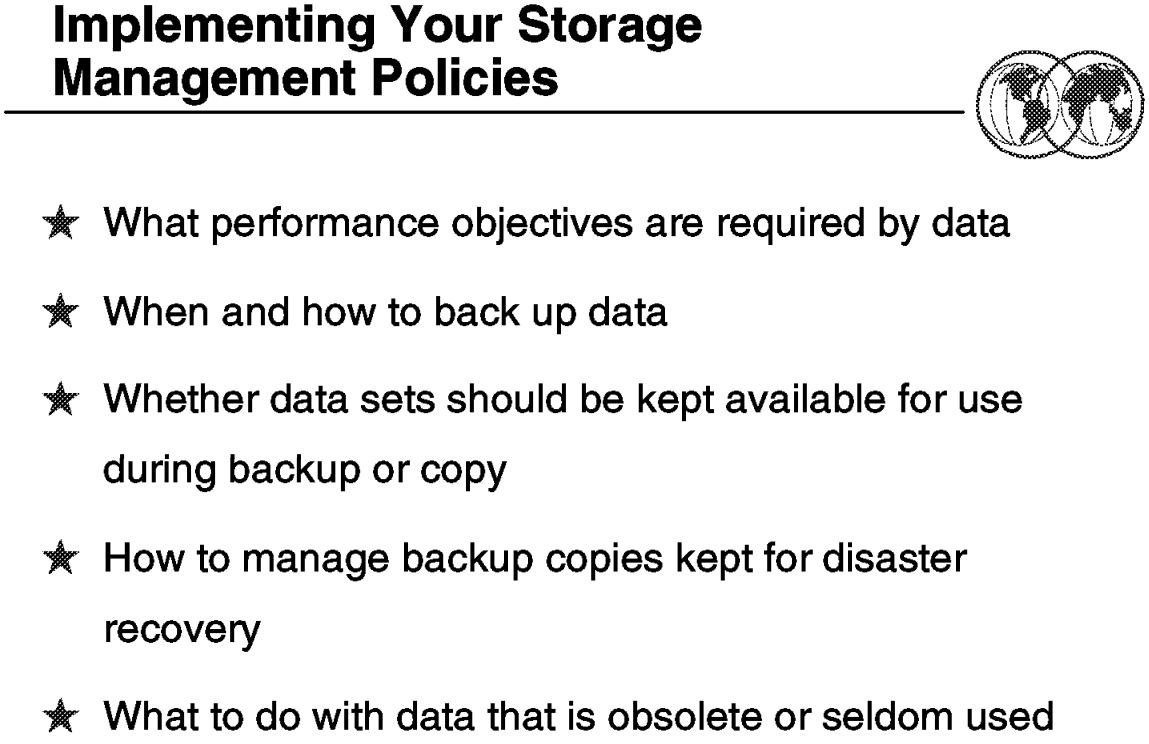


Figure 96. Implementing your storage management policies

2.4 Implementing your storage management policies

The purpose of a backup plan is to ensure the prompt and complete recovery of data. A well-documented plan identifies data that requires backup, the levels required, responsibilities for backing up the data, and methods to be used.

The policies defined by your installation represent decisions about your resources, such as:

- What performance objectives are required by the transactions accessing the data; based on these objectives you can try to better exploit cache data striping. By tracking data set I/O activities, you can make better decisions about data set caching policies and improve overall system performance. For object data, you can track transaction activities to monitor and improve OAM′s performance.
- When and how to back up data.emdash.do it incremental or total. Determine the backup frequency, the number of backup versions, and the retention period by consulting user group representatives. Be sure to consider whether certain data backups need to be synchronized. For example, if the output data from application A is used as input for application B, you must coordinate the backups of both applications to prevent logical errors in the data when they are recovered.
- Whether data sets should be kept available for use during backup or copy. You can store backup data sets on DASD or tape (this does not apply to objects). Your choice depends on how fast the data needs to be recovered, media cost, operator cost, floor space, power requirements, air conditioning, the size of the data sets, and whether you want the data sets to be portable.

- How to manage backup copies kept for disaster recovery, locally or in a vault. Related data sets should be backed up in aggregated tapes. Each application should have its own self-contained aggregate of data sets. If certain data sets are shared by two or more applications, you might want to ensure application independence for disaster recovery by backing up each application that shares the data. This is especially important for shared data in a distributed environment.
- What to do with data that is obsolete or seldom used. Data is obsolete when it has exceeded its expiration dates and is no longer needed. Some examples are old masters, listings, and permanent work files. To select obsolete data for deletion using DFSMSdss, issue the DUMP command and the DELETE parameter, and force OUTDDNAME to DUMMY.

Implementing and Monitoring Storage Management Policies

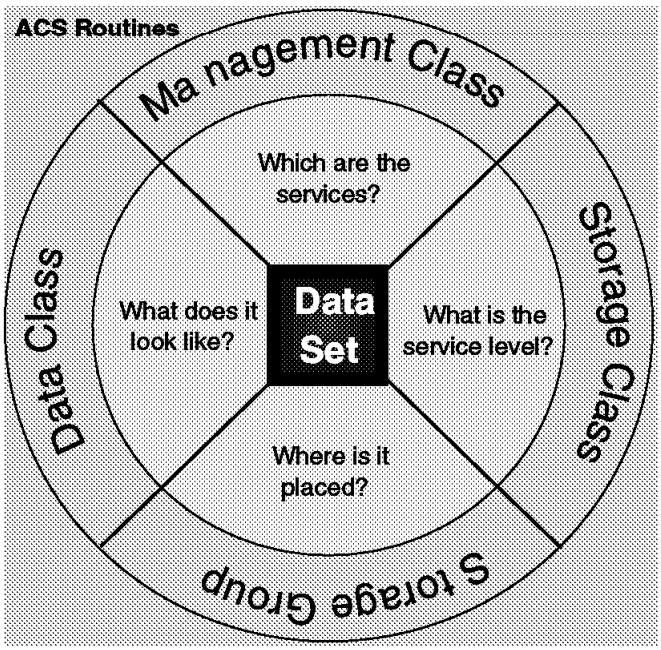


Figure 97. Implementing monitoring storage policies

2.5 Implementing and monitoring storage management policies

To implement a policy for managing storage, your storage administrator defines classes of space management, performance, and availability requirements for data sets at your installation. For example, the administrator can define one storage class for data entities requiring high performance and another for those requiring standard performance. Then, the administrator writes Automatic Class Selection (ACS) routines that use naming conventions, or other criteria of your choice, to automatically assign the classes that have been defined to data as that data is created. These ACS routines can then be validated and tested. When the ACS routines are started and the classes (also referred to as constructs) are assigned to the data, SMS uses the policies defined in the classes to apply to the data for the life of the data. Additionally, devices with various characteristics can be pooled together into storage groups, so that new data can be automatically placed on devices that best meet the needs for the data.

DFSMS/MVS facilitates all of these tasks by providing menu-driven, fill-in-the-blank panels with the Interactive Storage Management Facility (ISMF). ISMF panels make it easy to define classes, test and validate ACS routines, and perform other tasks to analyze and manage your storage. Note that many of these functions are available in batch through the NaviQuest tool.

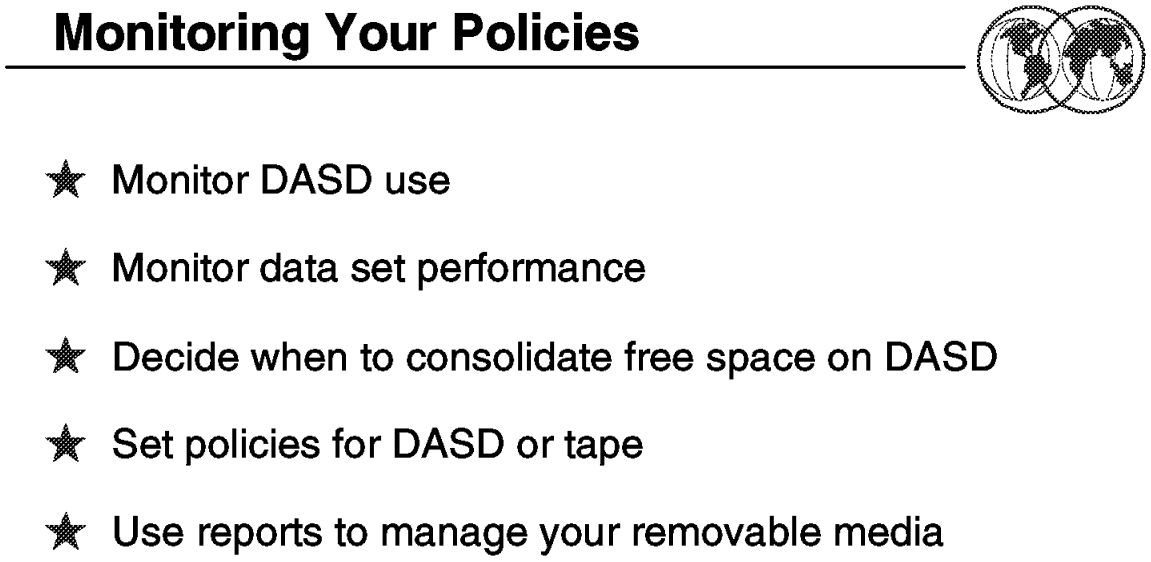


Figure 98. Monitoring your policies

2.5.1 Monitoring your policies

After you have established your installation's service levels and implemented policies based on those levels, you can use DFSMS/MVS facilities to see if your objectives have been met. Information on past use can help you develop more effective storage administration policies and manage growth effectively. Use the DFSMS/MVS Optimizer feature to help you monitor, analyze, and tune your policies. This visual shows the actions available to you.

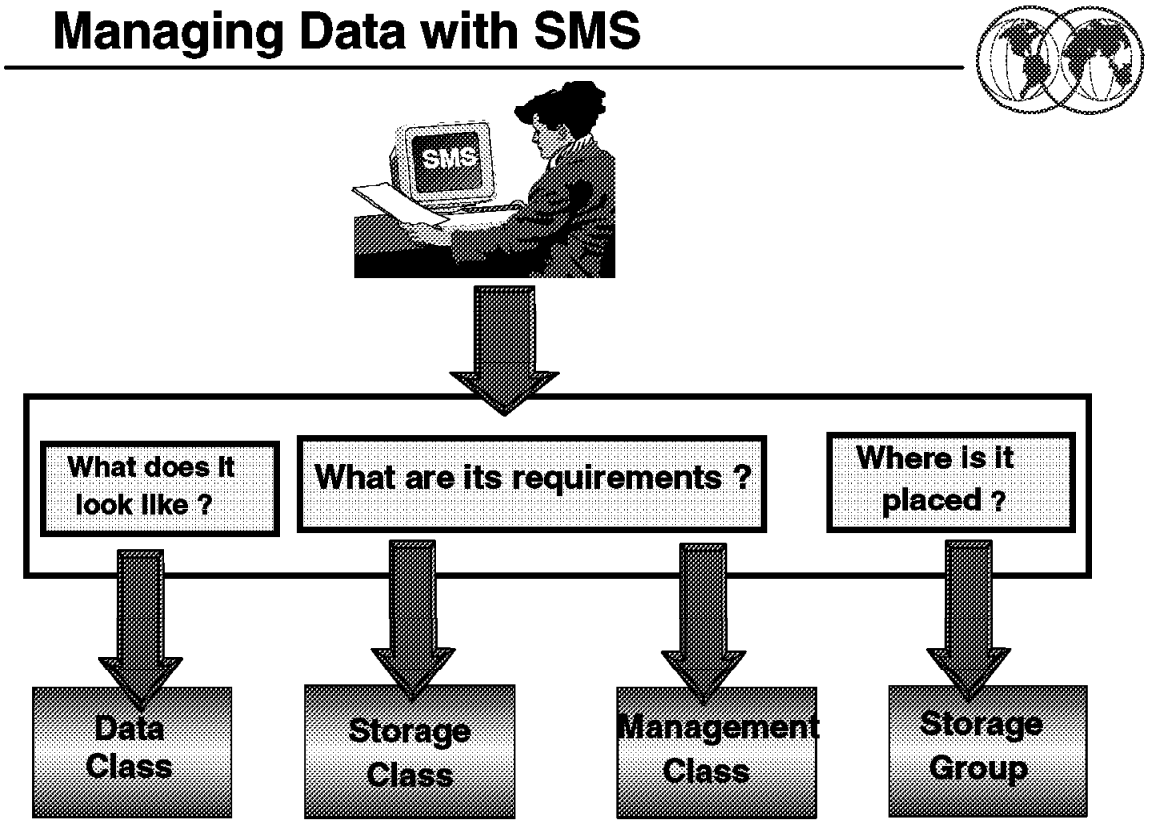


Figure 99. Managing data with SMS

2.6 Managing data with SMS

In the DFSMS environment, you use SMS classes and groups to set service requirements, performance goals, and data definition models for your installation. You use the Interactive Storage Management Facility (ISMF) to create the appropriate classes and groups, and Automatic Class Selection (ACS) routines to assign them (classes and groups) to data according to your installation's policies.

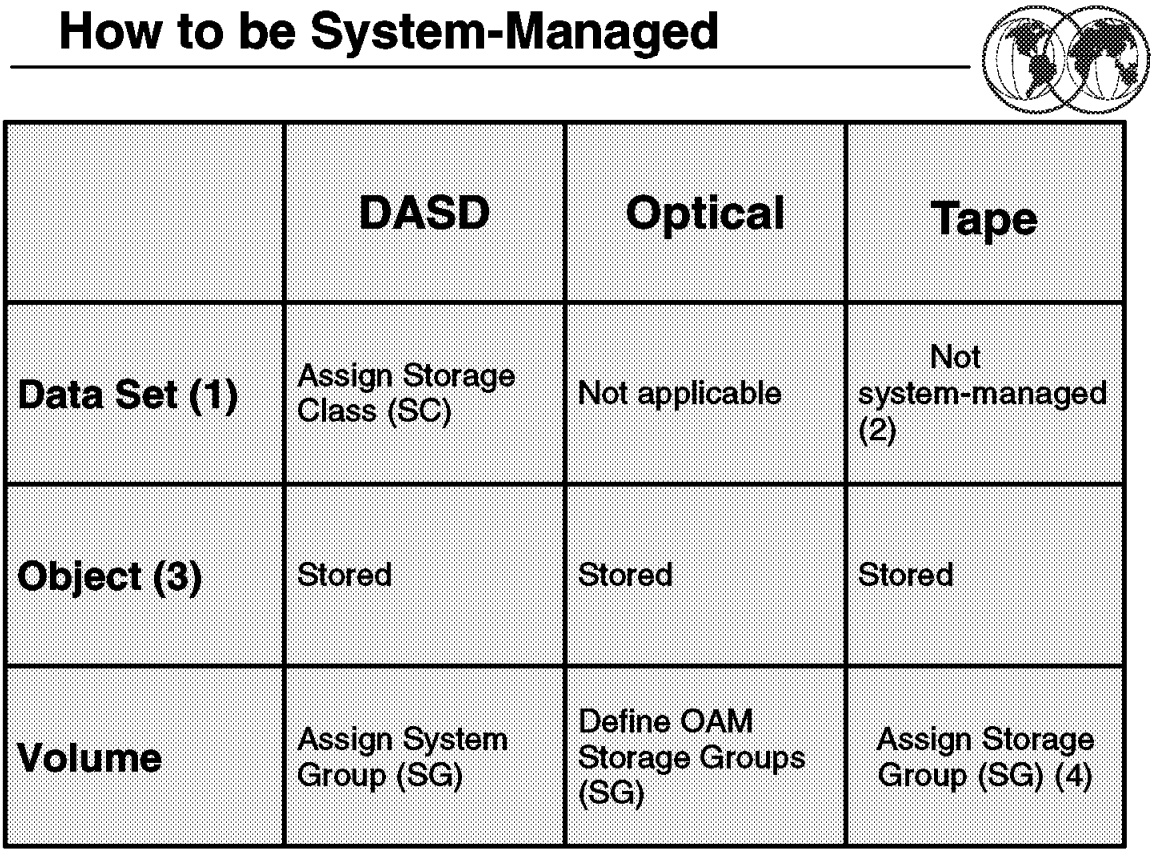


Figure 100. How to be system-managed

2.6.1 How to be system-managed

On systems that do not use DFSMS, storage management consists mostly of manual operations performed on individual data sets, and manual and automated operations performed on volumes. With SMS, you can automate storage management for individual data sets and objects, and for DASD, optical, and tape volumes. You use SMS classes and groups to define the goals and requirements that the system should meet for a data set or object. You use:

- Data class to define model allocation characteristics for data sets
- Storage class to define performance and availability goals
- Management class to define backup and retention requirements
- Storage group to create logical groupings of volumes to be managed as a unit

This visual shows how a data set, object, DASD volume, tape volume, or optical volume becomes system-managed. The numbers shown in parentheses are associated with the following notes:

1. A DASD data set is system-managed if you assign it a storage class. If you do not assign a storage class, the data set is directed to a non-system-managed DASD or tape volume—one that is not assigned to a storage group.
2. You can assign a storage class to a tape data set to direct it to a system-managed tape volume. However, only the tape volume is considered system-managed, not the data set.
3. Objects are also known as byte-stream data, this data is used in specialized applications such as image processing, scanned correspondence, and seismic measurements. Object data typically has

no internal record or field structure and, once written, the data is not changed or updated. However, the data can be referenced many times during its lifetime. Objects are processed by OAM. Each object has a storage class; therefore, objects are system-managed. The optical or tape volume on which the object resides is also system-managed.

4. Tape volumes are added to tape storage groups in tape libraries when the tape data set is created.

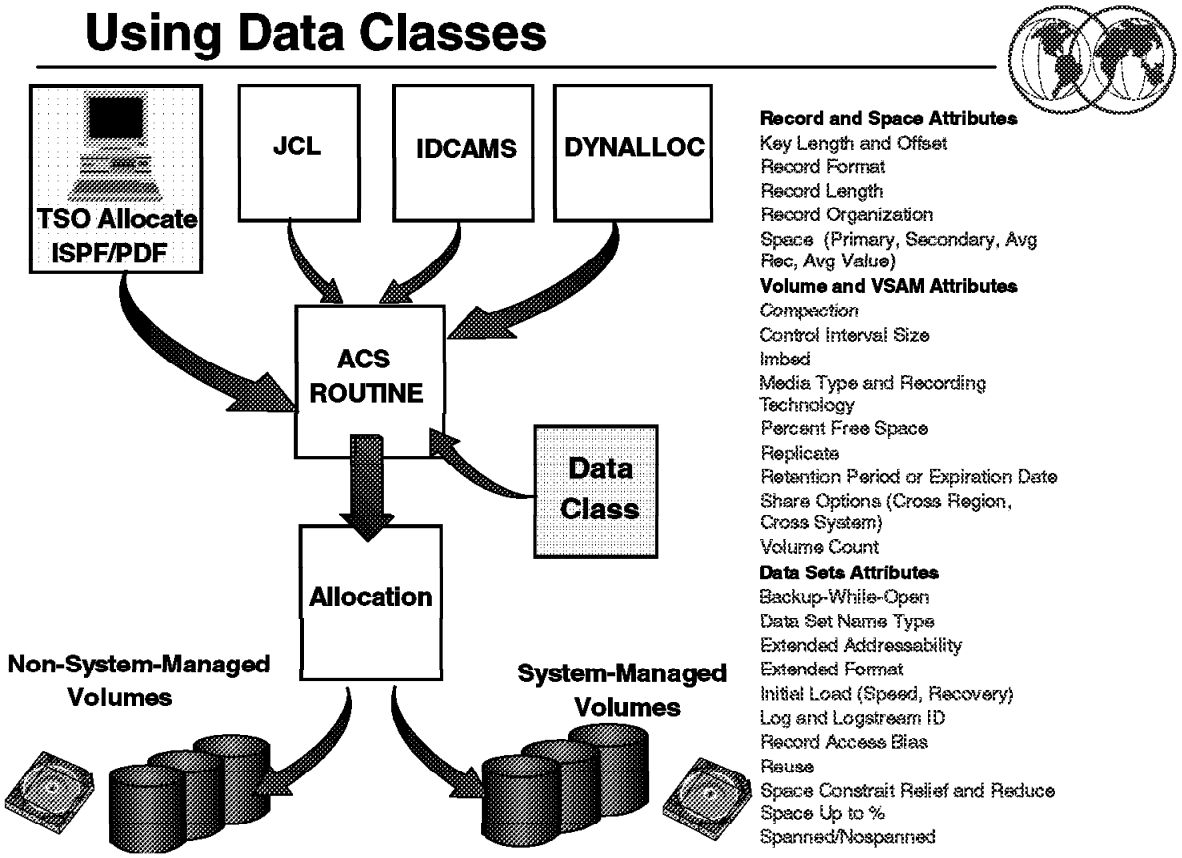


Figure 101. Using data classes

2.6.2 Using data classes

A data class is a collection of *allocation and space attributes* that you define. It is used when data sets are created. You can simplify data set allocation for your users by defining data classes that contain standard data set allocation attributes. You can use data classes with both system-managed and non-system-managed data sets, but some data class characteristics are only available with system-managed requests.

Data class attributes define space and data characteristics of data sets that are normally specified on JCL DD statements, TSO/E ALLOCATE commands, access method services (IDCAMS) DEFINE commands, dynamic allocation requests, and ISPF/PDF panels. For tape data sets, data class attributes can also specify the type of cartridge and recording method, and if the data is to be compacted. Users then need only specify the appropriate data classes to create standardized data sets.

You can use a data class to specify that a sequential data set should be striped. In this case, you must allocate extended format sequential data sets across DASD volumes to sustain the data rate you specify.

You can also use the data class automatic class selection (ACS) routine to automatically assign data classes to new data sets. For example, data sets with the low-level qualifiers LIST, LISTING, OUTLIST, or LINKLIST are usually utility output data sets with similar allocation requirements, and can all be assigned the same data class. Another way to attribute a data class to a data set is using the DC= parameter in the DD card of the referred data set.

This visual shows how data sets can be assigned a data class during data set creation.

Even though data class is optional, we usually recommend that you assign data classes to system-managed and non-system-managed data. Although the data class is not used after the initial allocation of a data set, the data class name is kept in the catalog entry for system-managed data sets for future reference. The data class name is not saved for non-system-managed data sets, although the allocation attributes in the data class are used to allocate the data set.

For objects on tape, we recommend that you do not assign a data class via the ACS routines. To assign a data class, specify the name of that data class on the SETOAM command.

If you change a data class definition, the changes only affect new allocations. Existing data sets allocated with the data class are not changed.

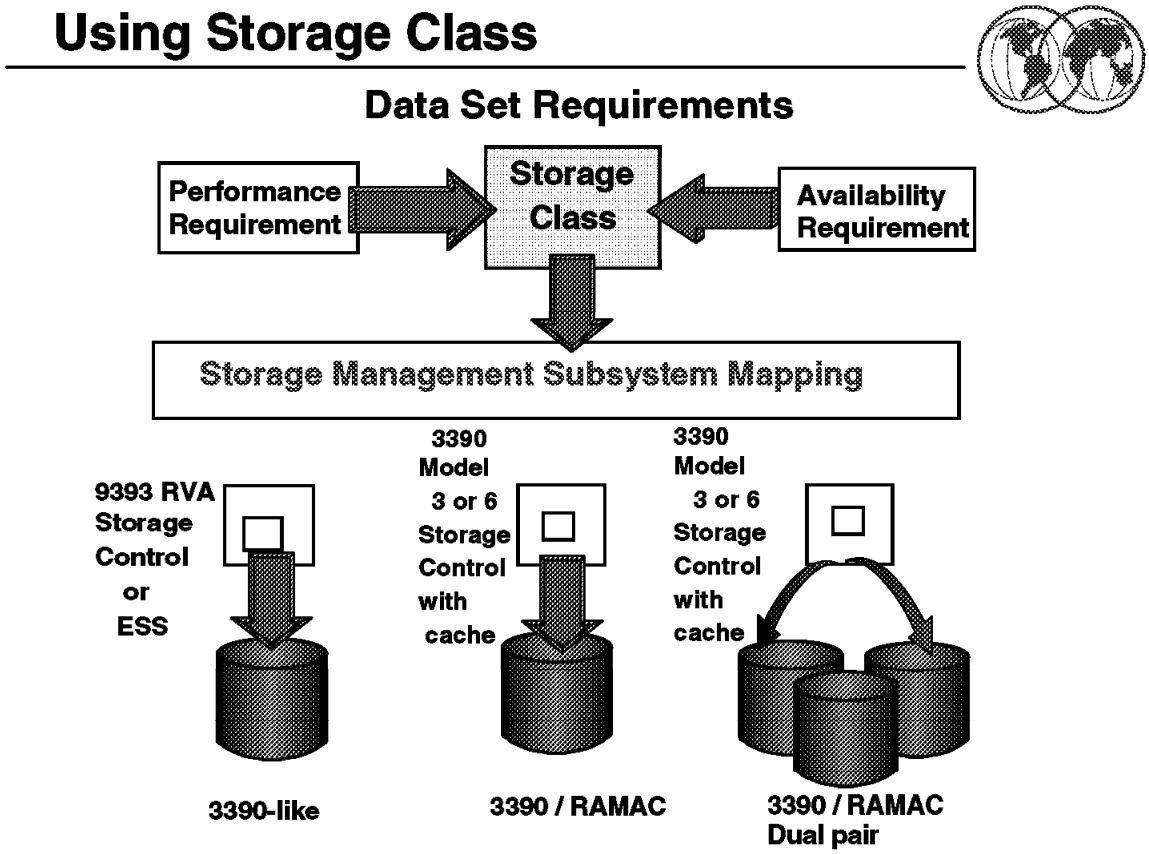


Figure 102. Using storage classes

2.6.3 Using storage classes

A storage class is a collection of *performance goals and availability* requirements that you define. It is used to select a device to meet those goals and requirements. Only system-managed data sets and objects can be assigned a storage class. Storage classes free users from having to know about the physical characteristics of storage devices and manually placing their data on appropriate devices.

Some of the availability requirements you can specify with storage classes can only be met by DASD volumes attached through a 3990-6, or 9390 and 9393 RVA storage control units. You can specify attributes that require the use of the dual copy, concurrent copy, remote copy, and snapshot features. This visual shows some storage control unit configurations and their storage class attribute values.

With storage class, you can assign a data set to dual copy volumes to ensure continuous availability for the data set. With dual copy, two current copies of the data set are kept on separate DASD volumes (by the control unit). If the volume containing the primary copy of the data set is damaged, the companion volume is automatically brought online and the data set continues to be available and current. Remote copy is the same, with the two volumes in distinct control units (generally remote).

You can use the ACCESSIBILITY attribute of the storage class to request that concurrent copy be used when data sets or volumes are backed up.

The 3990-6 and 9390 concurrent copy function enables you to take point-in-time copies of data by using a cache sidefile that is loaded with the time-zero version of the data. Time-zero refers to the state of the data when the concurrent copy session is started and before it gets updated. Before a record is

updated, it is copied to the cache sidefile thus creating a *before update* version or time-zero version of the record.

You can specify an I/O response time objective with storage class by using the millisecond response time (MSR) parameter. During data set allocation, the system attempts to select the closest available volume to the specified performance objective. Also along the data set life, through the use MSR, DFSMS dynamically uses the cache algorithms as DASD Fast Write (DFW) and Inhibit Cache Load (ICL) in order to reach the MSR target I/O response time. This DFSMS function is called dynamic cache management.

You can also use the storage class automatic class selection (ACS) routine to automatically assign data classes to new data sets. Another way to attribute a data class to a data set is by using the SC= parameter in the DD card that refers to the data set.

For objects, the system uses the performance goals you set in the storage class to place the object on DASD, optical, or tape volumes. The storage class is assigned to an object when it is stored or when the object is moved. The ACS routines can override this assignment.

If you change a storage class definition, the changes affect the performance service levels of existing data sets that are assigned to that class when the data sets are subsequently opened. However, the definition changes do not affect the location or allocation characteristics of existing data sets.

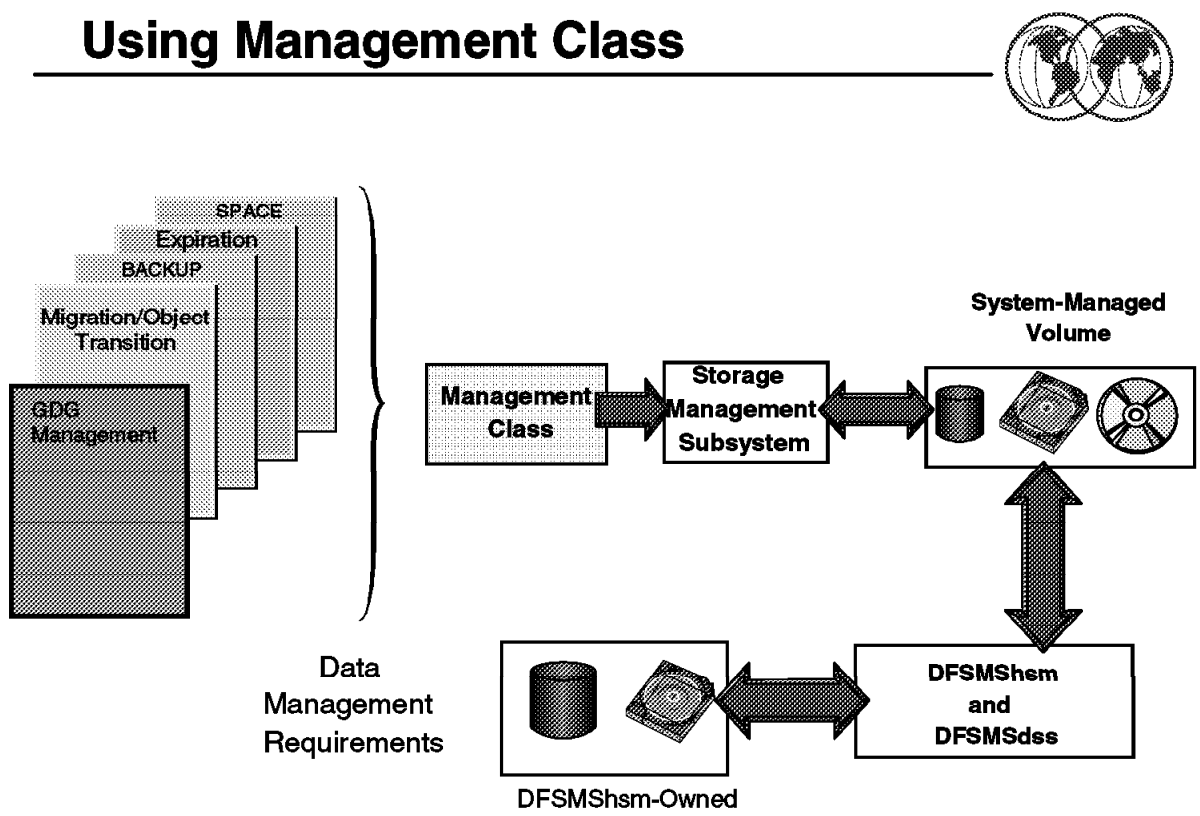


Figure 103. Using management class

2.6.4 Using management classes

A management class is a collection of *management attributes* that you define. It is used to control the retention, migration, backup, and release of allocated but unused space for data sets, or to control the retention, backup, and class transition of objects. Management classes let you define management requirements for individual data sets, rather than defining the requirements for entire volumes. All the data set functions described in the management class are executed by DFSMShsm and DFSMSdss programs.

You can also use the management class automatic class selection (ACS) routine to automatically assign data classes to new data sets. Another way to attribute a data class to a data set is using the MC= parameter in the DD card that refers to the data set.

If you do not explicitly assign a management class to a system-managed data set or object, the system uses the default management class. You can define your own default management class when you define your SMS base configuration.

For objects, you can:

- Assign a management class when it is stored, or
- Assign a new management class when the object is moved, or
- Change the management class by using the OAM Application Programming Interface (OSREQ CHANGE function)

The ACS routines can override this assignment for objects.

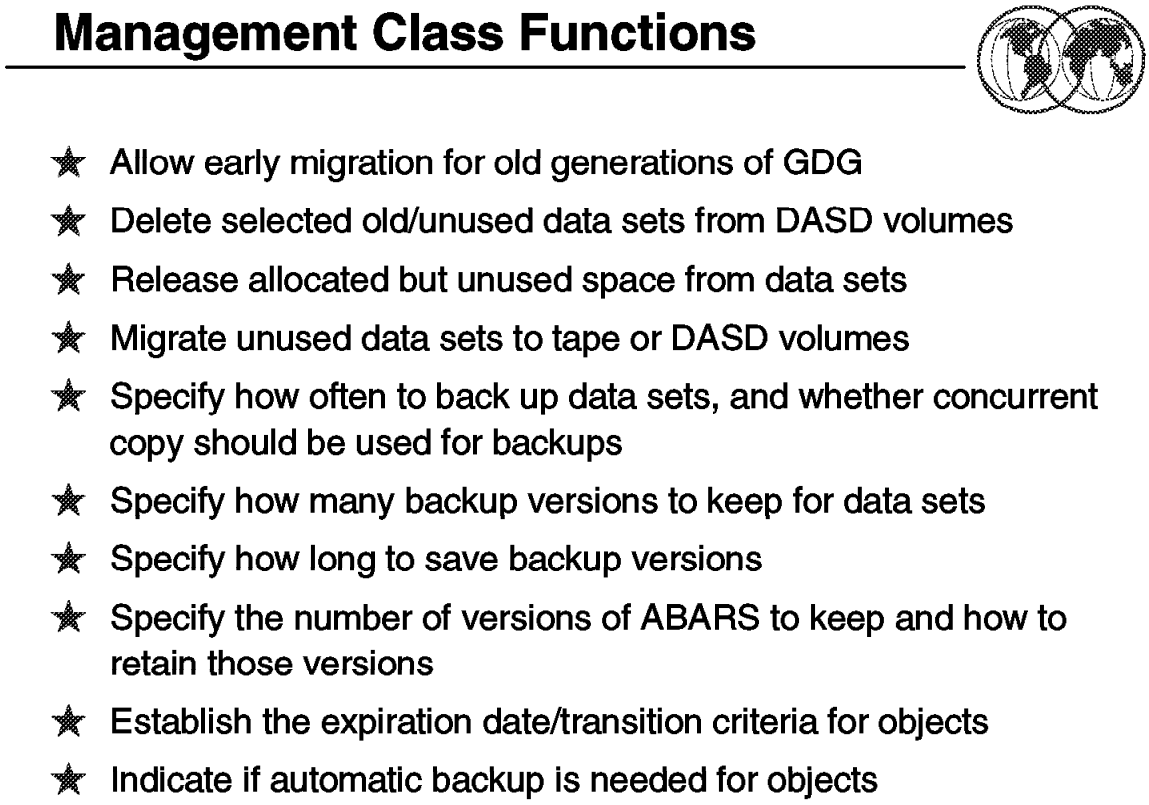


Figure 104. Management class functions

2.6.5 Management class functions

This visual shows the sort of functions an installation may define in a management class.

By classifying your data according to management requirements, you can define unique management classes to fully automate your data set and object management. For example, you can use management classes to control the migration of CICS user databases, DB2 user databases and archive logs, test systems and their associated data sets, and IMS archive logs. You can specify that DB2 image copies, and IMS image copies and change accumulation logs, be written to primary volumes and then migrated directly to migration level 2 tape volumes.

For objects, you use the class transition attributes to define when an object is eligible for a change in its performance objectives or management characteristics. For example, after a certain number of days you might want to move an object from a high-performance DASD volume to a slower optical volume. You can also use the management class to specify that the object should have a backup copy made when the OAM Storage Management Component (OSMC) is executing.

If you change a management class definition, the changes affect the management requirements of existing data sets and objects that are assigned to that class.

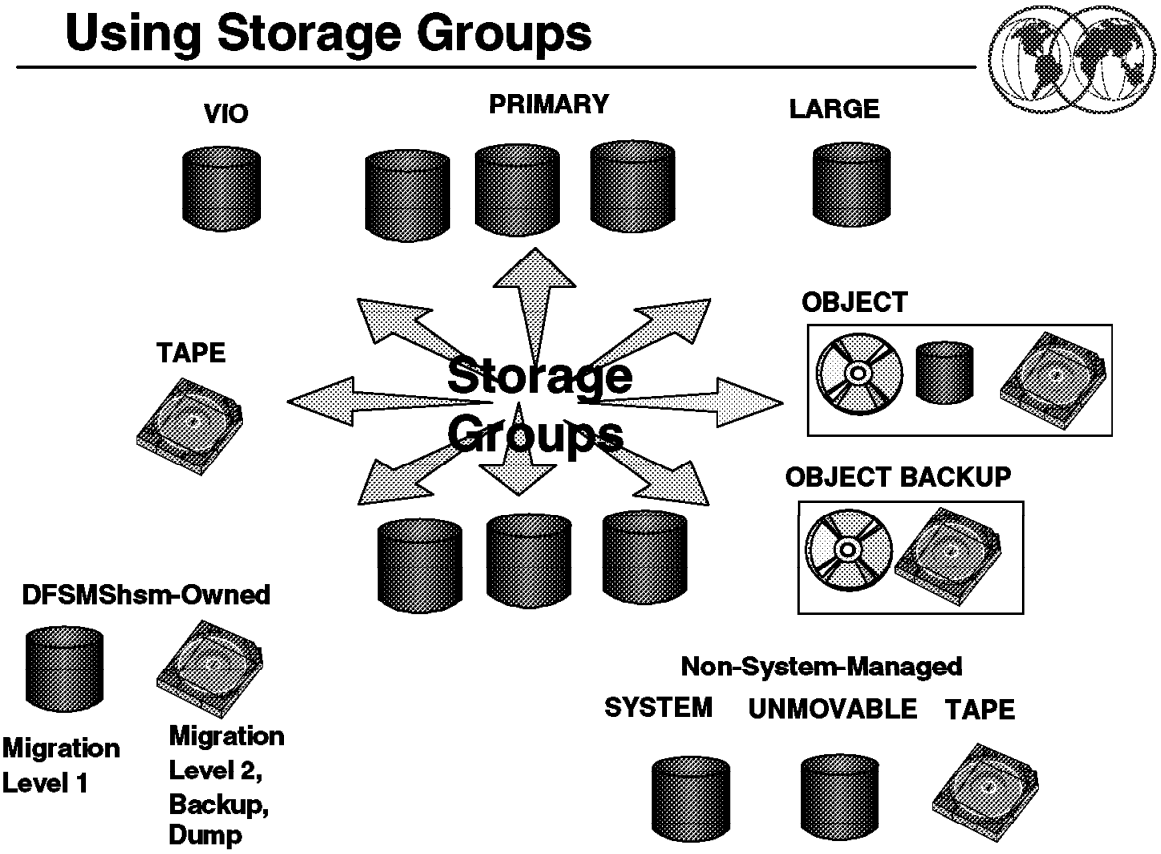


Figure 105. Using storages groups

2.6.6 Using storage groups

A storage group is a collection of storage volumes and attributes that you define. The collection can be a group of:

- System paging volumes
- DASD volumes
- Tape volumes
- Optical volumes
- Combination of DASD and optical volumes that look alike
- DASD, tape, and optical volumes treated as a single object storage hierarchy

Storage groups, along with storage classes, help reduce the requirement for users to understand the physical characteristics of the storage devices which contain their data.

You can direct new data sets to as many as 15 storage groups, although only one storage group is selected for the allocation. The system uses the storage class attributes, volume and storage group SMS status, MVS volume status, and available free space to determine the volume selected for the allocation. In a tape environment, you can also use tape storage groups to direct a new tape data set to an automated or manual tape library.

DFSMShsm uses some of the storage group attributes to determine if the volumes in the storage group are eligible for automatic space or availability management. This visual is an example of using

storage groups to group storage volumes for specific purposes. In this example, DASD volumes are grouped so that primary data sets, large data sets, DB2 data, IMS data, and CICS data are all separated. The VIO storage group uses system paging volumes for small temporary data sets. The tape storage groups are used to group tape volumes that are held in tape libraries. The object storage group can span optical, DASD, and tape volumes; the object backup storage group can contain either optical or tape volumes within one OAM invocation. Some volumes are not system-managed, and other volumes are owned by DFSMShsm for use in data backup and migration. DFSMShsm migration level 2 tape cartridges can be system-managed if you assign them to a tape storage group.

Unlike data, storage, and management classes, users cannot specify a storage group when they allocate a data set, although they can specify a unit and volume. Whether or not you honor their unit and volume request is an installation decision, but we recommend you discourage users from directly requesting specific devices. It is more effective for your users to specify the logical storage requirements of their data by storage and management class, which you can then verify in the automatic class selection routines.

However, if at allocation time there are more than one candidate device to receive the data set, SRM is still asked about the best candidate, performance-wise.

For objects, there are two types of storage groups, OBJECT and OBJECT BACKUP. An OBJECT storage group is assigned by OAM when the object is stored; the storage group ACS routine can override this assignment. There is only one OBJECT BACKUP storage group and all backup copies of all objects are assigned to this storage group.

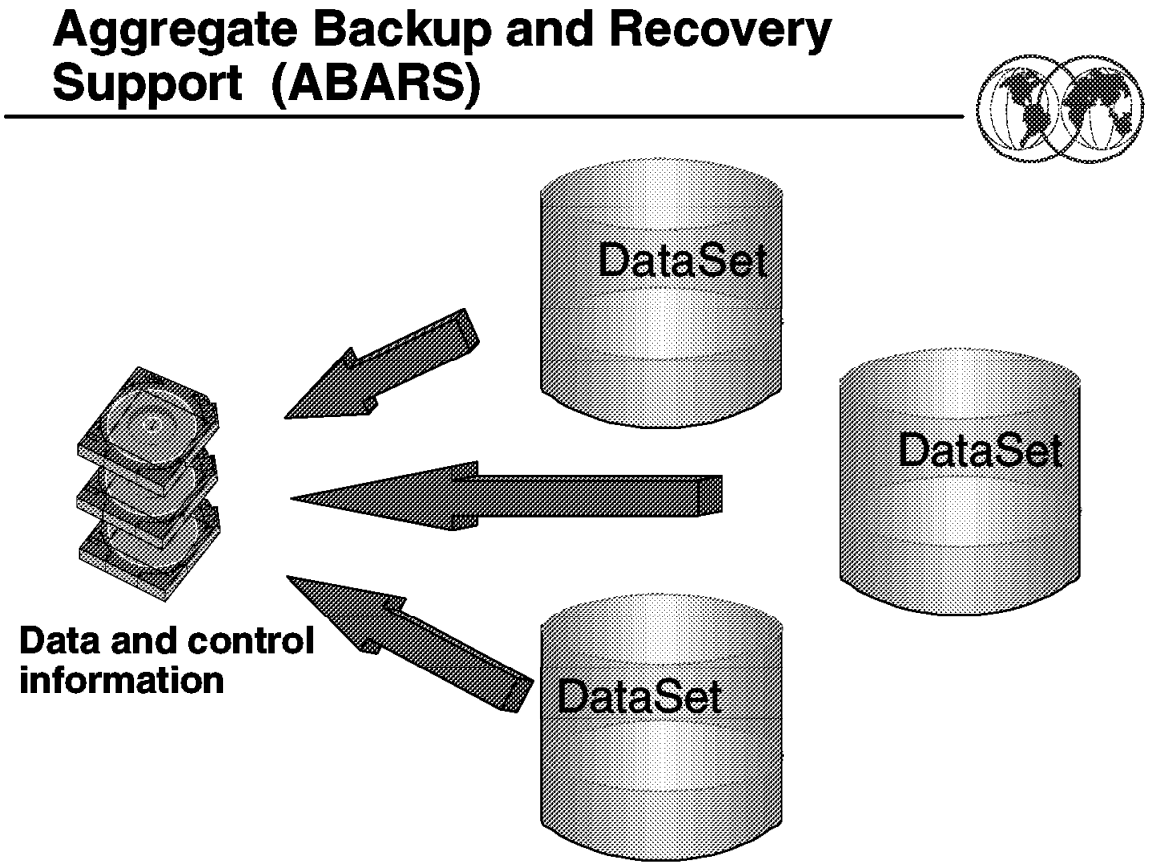


Figure 106. Aggregate backup and recovery support

2.6.7 Using aggregate backup and recovery support (ABARS)

Aggregate backup-and-recovery support, also called application backup application recovery support, is a command-driven process to back up and recover any user-defined group of data sets that are vital to your business. Then, an aggregate group is a collection of related data sets and control information that have been pooled to meet a defined backup or recovery strategy. If a disaster occurs, you can use these backups at a remote or local site to recover critical applications.

The user-defined group of data sets can be those belonging to an application or any combination of data sets that you want treated as a separate entity. Aggregate processing enables you to:

- Back up and recover data sets by application, to enable business to resume at a remote site if necessary
- Move applications in a non-emergency situation in conjunction with personnel moves or workload balancing
- Duplicate a problem at another site

You can use aggregate groups as a supplement to using management class for applications that are critical to your business. You can associate an aggregate group with a management class. The management class specifies backup attributes for the aggregate group, such as the copy technique for backing up DASD data sets on primary volumes, the number of aggregate versions to retain, and how long to retain versions. Aggregate groups simplify the control of backup and recovery of critical data sets and applications.

Although SMS must be used on the system where the backups are performed, you can recover aggregate groups to systems that are not using SMS, provided that the groups do not contain data which requires that SMS be active, such as PDSEs. You can use aggregate groups to transfer applications to other data processing installations or migrate applications to newly-installed DASD volumes. You can transfer the application's migrated data, along with its active data, without recalling the migrated data.

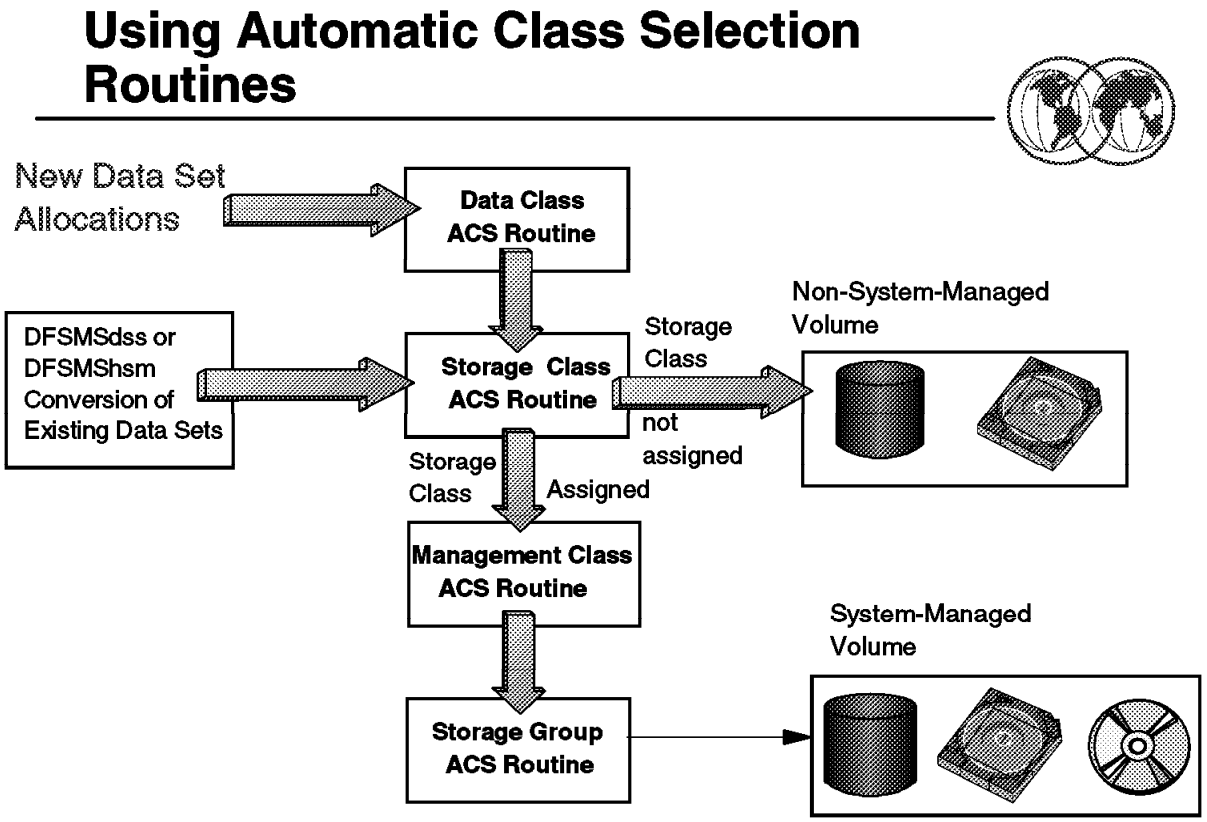


Figure 107. Using automatic class selection

2.6.8 Using automatic class selection routines

You use automatic class selection (ACS) routines to assign classes (data, storage, and management) and storage group definitions to data sets, database data, and objects. You write ACS routines using the ACS language, which is a high-level programming language. Once written, you use the ACS translator to translate the routines to object form so they can be stored in the SMS configuration.

The ACS language contains a number of read-only variables, which you can use to analyze new data allocations. For example, you can use the read-only variable &DSN to make class and group assignments based on data set or object collection name, or &LLQ to make assignments based on the low-level qualifier of the data set or object collection name. You cannot alter the value of read-only variables.

You use the four read-write variables to assign the class or storage group you determine for the data set or object, based on the routine you are writing. For example, you use the &STORCLAS variable to assign a storage class to a data set or object.

For a detailed description of the ACS language and its variables, see *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920.

For each SMS configuration, you can write as many as four routines: one each for data class, storage class, management class, and storage group. Use ISMF to create, translate, validate, and test the routines.

The visual shows the order in which ACS routines are processed. Data can become system-managed if the storage class routine assigns a storage class to the data, or if it allows a user-specified storage class to be assigned to the data. If this routine does not assign a storage class to the data, the data cannot reside on a system-managed volume.

Because data allocations, whether dynamic or through JCL, are processed through ACS routines, you can enforce installation standards for data allocation on system-managed and non-system-managed volumes. ACS routines also enable you to override user specifications for data, storage, and management class, and requests for specific storage volumes.

You can use the ACS routines to determine the SMS classes for data sets created by the Distributed FileManager/MVS. If a remote user does not specify a storage class, and if the ACS routines decide that the data set should not be system-managed, the Distributed FileManager/MVS terminates the creation process immediately and returns an error reply message to the source. Therefore, when you construct your ACS routines, consider the potential data set creation requests of remote users.

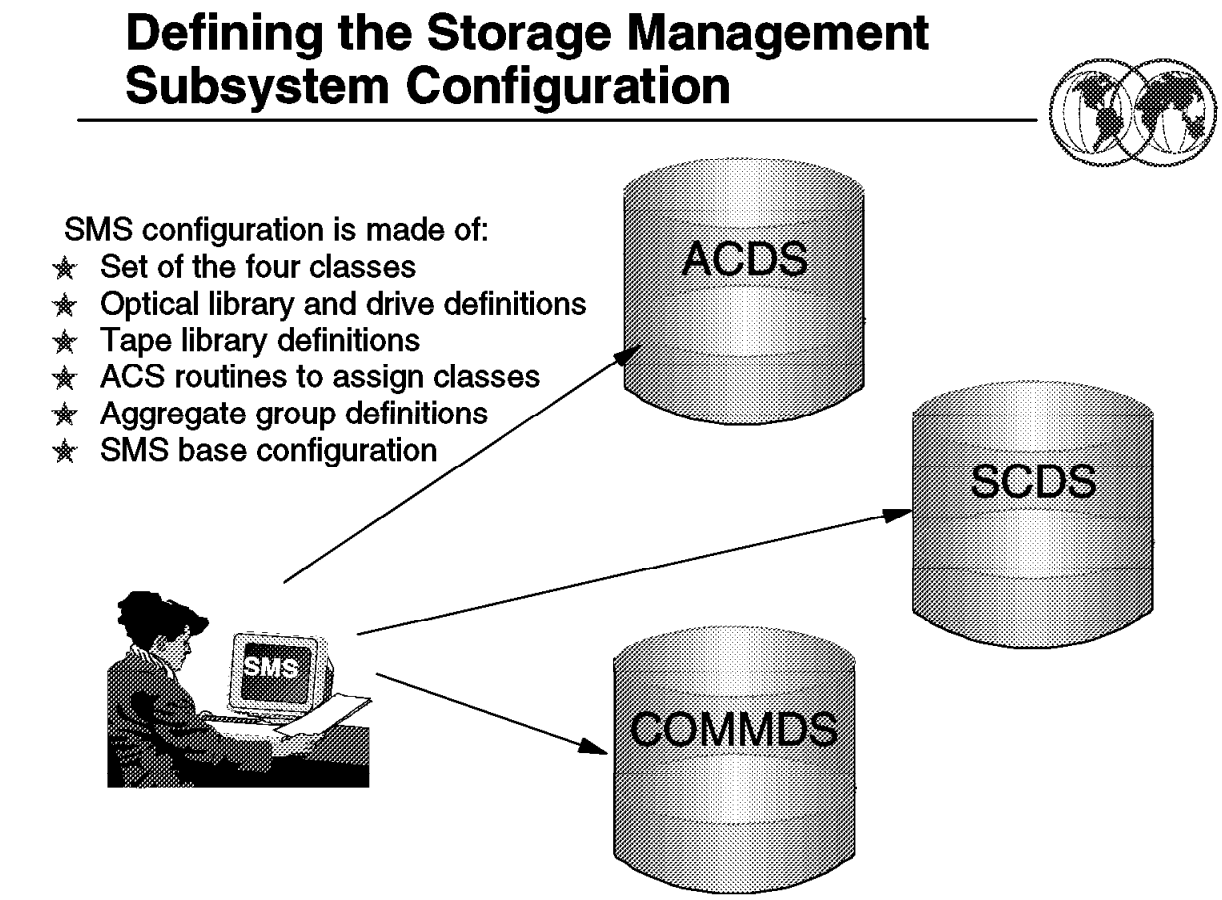


Figure 108. Defining the storage management subsystem.

2.7 Defining the storage management subsystem configuration

An SMS configuration is composed of a set of data class, management class, storage class, storage group, optical library and drive definitions, tape library definitions, and ACS routines to assign the classes and groups. It also includes the aggregate group definitions and the SMS base configuration. The SMS base configuration contains default information such as default management class and default device geometry. It also identifies the systems in the installation for which the subsystem manages storage.

This information is stored in SMS control data sets, which are VSAM linear data sets. You can define these control data sets using the access method services `DEFINE CLUSTER` command.

You must define the control data sets before activating SMS. Although you need only allocate the data sets from one system, the active control data set (ACDS) and communications data set (COMMDS) must reside on a device that can be accessed by every system to be managed with the SMS configuration.

Here we will introduce the concept of SMSplex. It means a set of OS/390 images sharing the same set of SMS control data sets (ACDS and COMMDS). It is recommended that your SMSplex matches your sysplex (parallel or basic)

SMS uses the following types of control data sets:

- Source Control Data Set (SCDS)
- Active Control Data Set (ACDS) ehp2.

- Communications Data Set (COMMDS)

2.7.1.1 Source Control Data Set (SCDS)

The SCDS contains a set of SMS classes and groups and translated ACS routines that implement a specific set of storage management policies.

This is the source control data set, the SMS configuration that you develop and test. You can have several SCDSs. One should correspond to your current policies. Retain at least one prior configuration should you need to regress to it because of error. The SCDS is never used to manage allocations.

2.7.1.2 Active Control Data Set (ACDS)

The ACDS is the system′s active copy of the current SCDS. When you activate a configuration, SMS copies the existing configuration from the specified SCDS into the ACDS. By using copies of the SMS classes, groups, volumes, optical libraries, optical drives, tape libraries, and ACS routines rather than the originals, you can change the current storage management policy without disrupting it. For example, while SMS uses the ACDS, you can:

- Create a copy of the ACDS
- Create a backup copy of an SCDS
- Modify an SCDS
- Define a new SCDS

The ACDS must reside on a shared device to ensure that all systems in the installation use the same active configuration.

We recommend that you have extra ACDSs in case a hardware failure causes the loss of your primary ACDS. It must reside on a shared device, accessible to all systems, to ensure that they share a common view of the active configuration. Do not have the ACDS reside on the same device as the COMMDS or SCDS. Both the ACDS and COMMDS are needed for SMS operation across the complex. Separation protects against hardware failure. You should also create a backup ACDS in case of hardware failure or accidental data loss or corruption.

2.7.1.3 Communications Data Set (COMMDS)

The COMMDS data set contains the name of the ACDS and storage group volume statistics. It enables communication between SMS systems in a multisystem environment. The COMMDS also contains space statistics, SMS status, and MVS status for each system-managed volume. Although only one COMMDS is used at a time for an SMS installation, we recommend that you have more COMMDSs on different volumes for recovery purposes.

The COMMDS must reside on a shared device accessible to all systems. However, do not allocate it on the same device as the ACDS. Create a spare COMMDS in case of a hardware failure or accidental data loss or corruption. SMS activation fails if the COMMDS is unavailable.

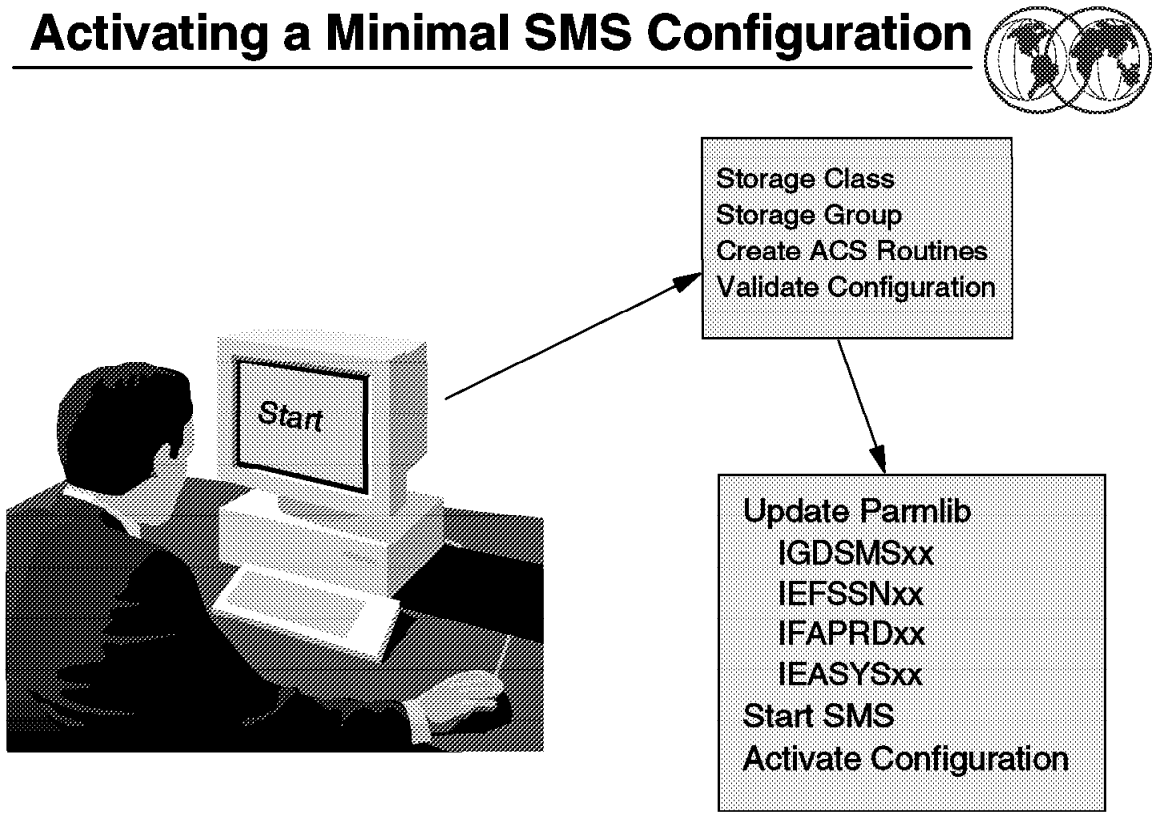


Figure 109. Activating a minimal SMS configuration

2.8 Activating a minimal SMS configuration

This section presents the steps necessary to define and activate a minimal SMS configuration. This establishes an operating environment for the storage management subsystem, without data sets becoming system-managed.

Activation of a minimal configuration lets you gain experience in managing an SMS configuration without affecting your JCL or data set allocations. During this phase, you can:

- Gain experience with ISMF applications for the storage administrator.
  - You use ISMF applications to define and activate a minimal SMS configuration.
- Gain experience with the new operator commands that control operation of resources controlled by SMS.
  - New MVS commands are used to activate an SMS configuration and to display and control storage group and volume status.
- Learn how the SMS base configuration can affect allocations for non-system-managed data sets. The base configuration for your minimal SMS contains installation defaults for data sets:
  - For non-system-managed data sets, you can specify default device geometry to ease the conversion from device-dependent space calculations to the device-independent method implemented by SMS.
  - For system-managed data sets, you can specify a default management class to be used for data sets that are not assigned a management class by your management class ACS routine.
- Use simplified JCL.
  - With the minimal configuration active, you can use simplified JCL.

- Implement allocation standards.
  - With the minimal configuration active, you can develop a data class ACS routine to enforce your standards.

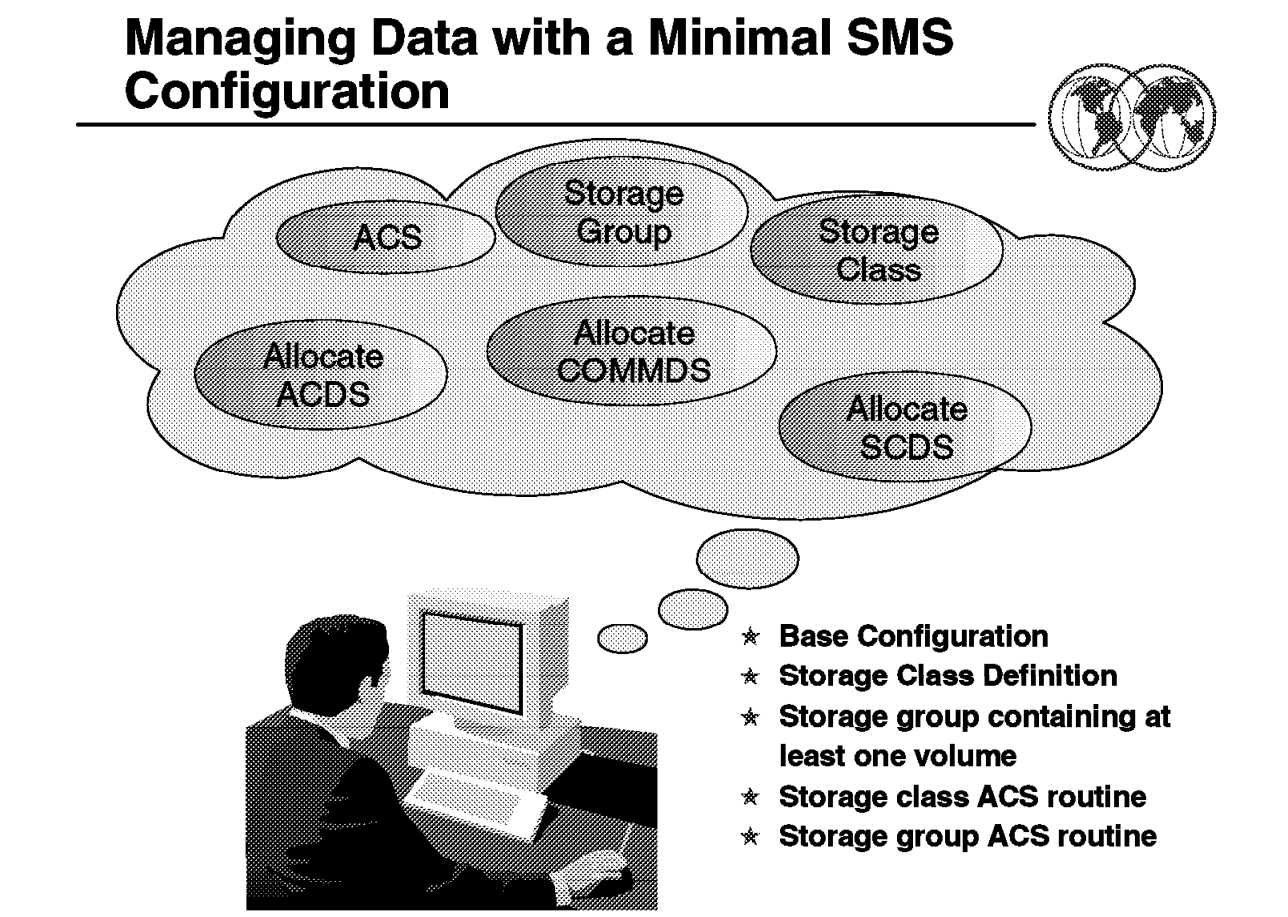


Figure 110. Managing data with a minimal SMS configuration

2.8.1 Managing data with a minimal SMS configuration

The visual shows the elements in the minimal SMS configuration:

- Base configuration
- Storage class definition
- Storage group containing at least one volume
- Storage class ACS routine
- Storage group ACS routine

All of these elements are required for a valid SMS configuration, except for the storage class ACS routine. However, we recommend that a storage class ACS routine be part of your minimal configuration. This prevents users from externally specifying a storage class on their DD statements (STORCLASS keyword), causing the data set to be system-managed before you are ready.

The storage class ACS routine ensures that the storage class read/write variable is always set to null. The storage group ACS routine is never run because no data sets are system-managed. So, no data sets are allocated as system-managed by the minimal configuration.

The base configuration consists of the names for each system or system group in the SMS complex, the default management class, and the default device geometry and unit information.

Users become familiar with the device-independent space allocation implemented by SMS facilities supported by the minimal configuration. Specifically, the base configuration contains a default unit that corresponds to a DASD esoteric (such as SYSDA). Default geometry for this unit is specified in bytes/track and tracks/cylinder for the predominant device type in the esoteric. If users specify the esoteric, or do not supply the UNIT parameter for new allocations, the default geometry converts space allocation requests into device-independent units, such as KBs and MBs. This quantity is then converted back into device-dependent tracks based on the default geometry.

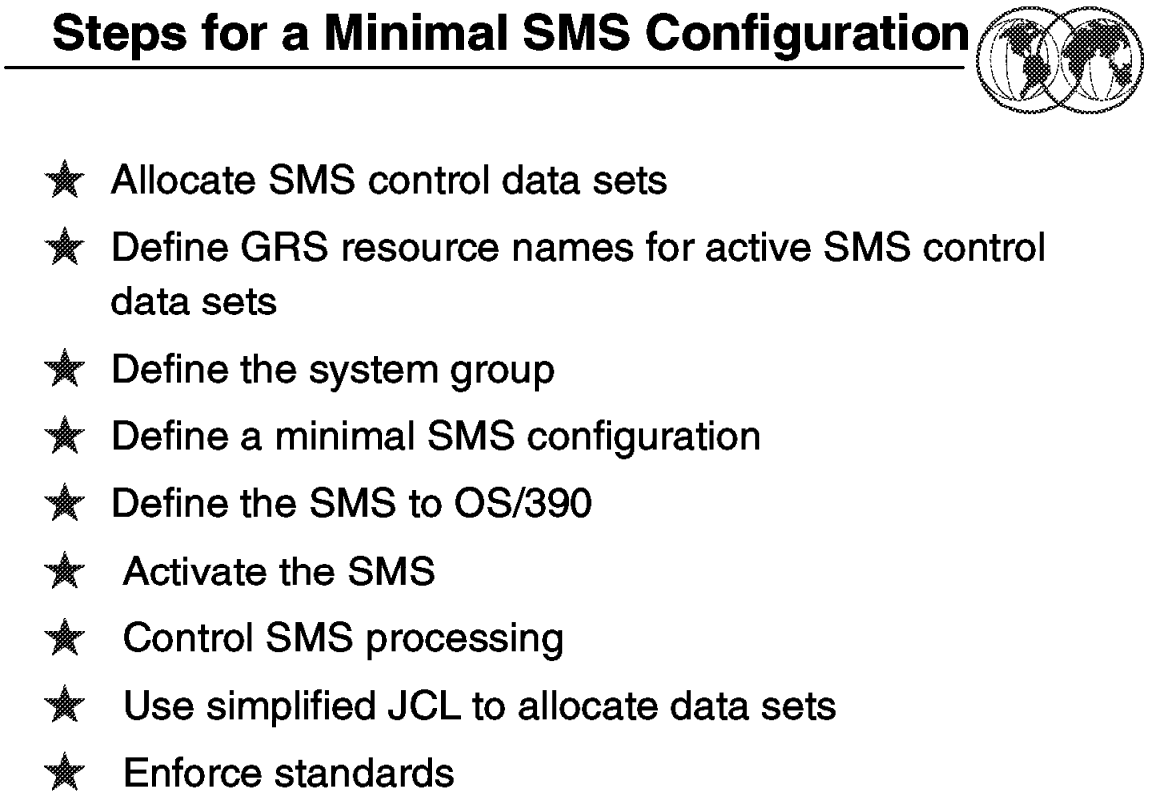


Figure 111. Steps for a minimal SMS configuration

2.8.2 Steps for a minimal SMS configuration

This visual shows the steps to activate the minimal configuration. These steps are covered in the next visuals.

To use SMS effectively, use the information in this chapter and that in the *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920, *MVS/ESA Storage Management Library Managing Data*, SC26-3124, and *MVS/ESA Storage Management Library Managing Storage Groups*, SC26-3125.

Allocating SMS Control Data Sets

```
//  EXEC  PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DEFINE CLUSTER(NAME(YOUR.OWN.SCDS) LINEAR VOLUME(D65DM1)      -
          TRK(25 5) SHAREOPTIONS(2,3))                         -
          DATA(NAME(YOUR.OWN.SCDS.DATA))

 DEFINE CLUSTER(NAME(YOUR.OWN.ACDS) LINEAR VOLUME(D65DM2)      -
          TRK(25 5) SHAREOPTIONS(3,3))                         -
          DATA(NAME(YOUR.ACDS.DATA))

 DEFINE CLUSTER(NAME(YOUR.OWN.COMMDS) LINEAR VOLUME(D65DM3)    -
          TRK(1 1) SHAREOPTIONS(3,3))                          -
          DATA(NAME(YOUR.OWN.COMMDS.DATA))
```

Figure 112. Allocating SMS control data sets

2.8.3 Allocating SMS control data sets

SMS stores its class and group definitions, translated ACS routines, and system information in the control data sets ACDS, COMMDS, and SCDS.

Calculating the SCDS and ACDS sizes

The size of the ACDS/SCDS may allow constructs for up to 32 systems. Be sure to allocate sufficient space for the ACDS and SCDS, since insufficient ACDS size can cause errors such as failing SMS activation. See *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920, for the formula used to calculate the appropriate SMS control data set size.

Calculating the COMMDS size

The size of the communications data set (COMMDS) increased in DFSMS/MVS 1.3, because the amount of space required to store system-related information for each volume increased. To perform a precise calculation of the COMMDS size, use the formula in *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920.

Defining the control data sets

After you have calculated their respective sizes, define the control data sets to MVS using access method services. The JCL in this visual is an example of how to define these data sets. Because

these data sets are allocated before SMS is activated, space is allocated in tracks. Allocations in KBs or MBs are only supported when SMS is active.

Specify SHAREOPTIONS(2,3) only for the SCDS. This lets one update-mode user operate simultaneously with other read-mode users between regions.

Specify SHAREOPTIONS(3,3) for the ACDS and COMMDS. These data sets must be shared between systems that are managing a shared DASD configuration in a DFSMS environment.

2.8.4 Define GRS resource names for active SMS control data sets

You should place resource name IGDCDSXS in the RESERVE conversion RNL as a generic entry. This will minimize delays due to contention for resource and prevent deadlocks associated with the VARY SMS command. For more information, see *DFSMS/MVS Planning for Installation*, SC26-3123.

Prior to DFSMS 1.5, if you should not have more than one SMSplex controlled by one GRSPlex. The name of the ENQ resource was not associated with the ACDS name, this caused performance problems due to false contention (different resource and same name). In DFSMS 1.5, the RNAME is appended by the BCDS name.

If there are multiple SMS complexes within a global resource serialization complex, be sure to use unique COMMDS and ACDS data set names to prevent false contention. For information on allocating COMMDS and ACDS data set names, see *DFSMS/MVS Implementing System-Managed Storage*, SC26-3123.

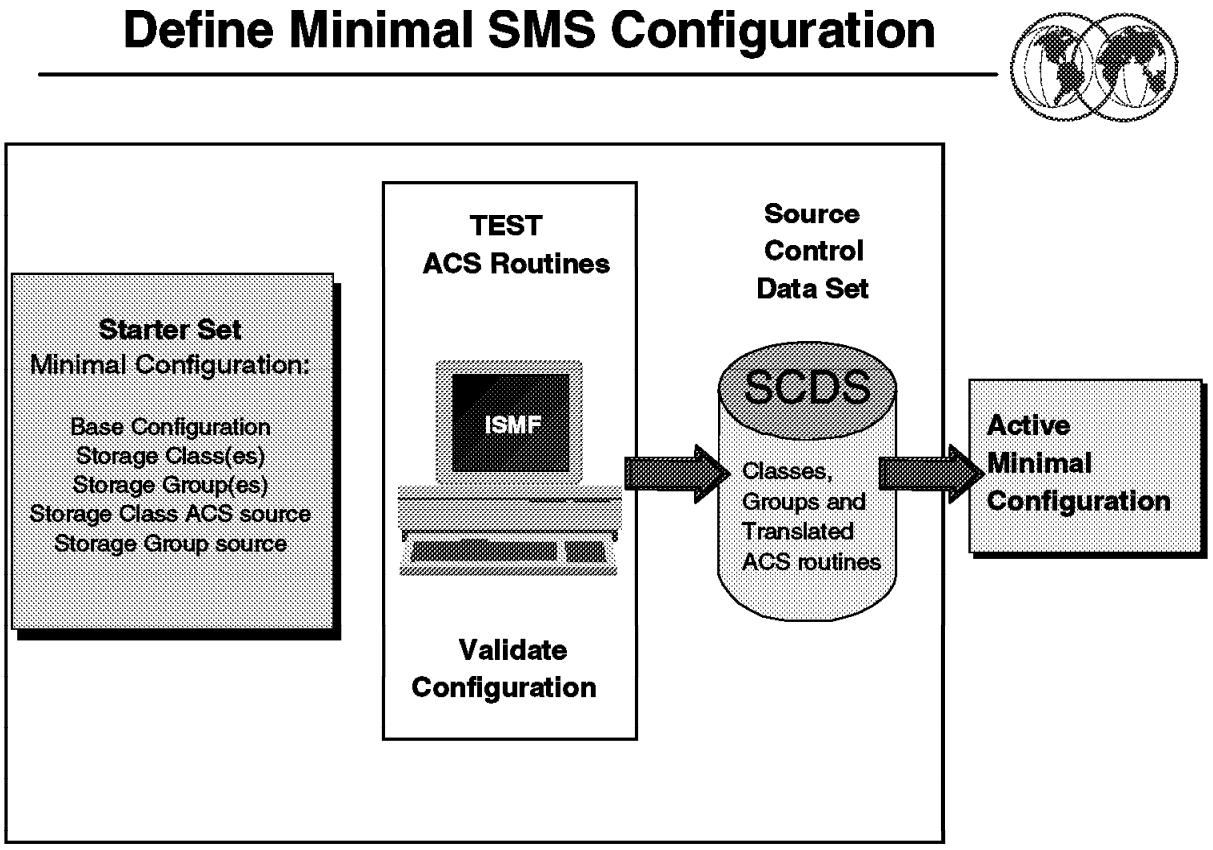


Figure 113. Defining a minimal SMS configuration

2.8.5 Defining a minimal SMS configuration

This visual shows the components of a minimal SMS configuration.

Note: The volume does not have to exist as long as you do not direct allocations to either the storage group or the volume.

The following sections describe the steps you should follow to define a minimal configuration.

2.8.5.1 Defining the SMS base configuration

Use ISMF to create your SCDS base. The starter set configuration can be used as a model for your own SCDS. For a detailed description of base configuration attributes and how to use ISMF to define its contents, see *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920.

2.8.5.2 Defining SMS classes and storage group

To define a minimal configuration, define a storage class and storage group, and create their respective ACS routines. Defining a data class and a management class and creating their respective ACS routines are not required for a valid SCDS. However, because of the importance of the *default management class*, we recommend that you include it in your minimal configuration. For a detailed description of SMS classes and groups, see *MVS/ESA Storage Management Library Managing Data*, SC26-3124. For detailed information on defining SMS classes and groups using ISMF, see *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920.

2.8.5.3 Defining the storage class

You must define at least one storage class name to SMS. Because a minimal configuration does not include any system-managed volumes, no performance or availability information need be contained in the minimal configuration's storage class. Specify an artificial storage class, NONSMS. This class is later used by the storage administrator to create non-system-managed data sets on an exception basis. In the storage class ACS routine, &STORCLAS is set to a null value to prevent users from coding a storage class in JCL before you want to have system-managed data sets.

You can define the class, NONSMS, in your configuration in one of two ways:

- Define the class using the DEFINE option of the ISMF storage class application
- Use the ISMF COPY line operator to copy the definition of NONSMS from the starter set's SCDS to your own SCDS

Defining the storage group

You must define at least one pool storage group name to SMS, and at least one volume serial number to this storage group (a storage group with no volumes defined is not valid). This volume serial number should be for a nonexistent volume to prevent the occurrence of JCL errors from jobs accessing data sets using a specific volume serial number.

Defining a non-existent volume lets you activate SMS without having any system-managed volumes. No data sets are system-managed at this time. This condition provides an opportunity to experiment with SMS without any risk to your data.

Define the NOVOLS storage group in your SCDS.

Defining the default management class

Define a default management class and name it STANDEF to correspond with the entry in the base configuration. We recommend that you specifically assign all system-managed data to a management class. If you do not supply a default, DFSMShsm uses two days on primary storage, and 60 days on migration level 1 storage, as the default.

No management classes are assigned when the minimal configuration is active. Definition of this default is done here to prepare for use in the managing permanent data milestone.

The management class, STANDEF, is defined in the starter set's SCDS. You can copy its definition to your own SCDS in the same way as the storage class, NONSMS, was copied.

Creating and writing ACS routines

After you define the SMS classes and group, develop your ACS routines. In the minimal configuration, you assign a null storage class in the storage class ACS routine. The storage group ACS routine is not run if a null storage class is assigned. However, you must code a trivial one to satisfy the SMS requirements for a valid SCDS. After you have written the ACS routines, use ISMF to translate them into executable form.

1. If you do not have the starter set, allocate a fixed-block PDS or PDSE with LRECL=80 to contain your ACS routines. Otherwise, go to the next step.
2. Enter a 7 (AUTOMATIC CLASS SELECTION) on the ISMF Primary Option Menu to display the ACS Application Selection panel.

Translating the ACS routines

The translation process checks the routines for syntax errors and converts the code into an ACS object. If the code translates without any syntax errors, then the ACS object is stored in the SCDS.

1. Select option 2 (TRANSLATE) from the ACS Application Selection panel and press Enter to display the Translate ACS Routines panel.

2.8.5.4 Validating the SCDS

When you validate your SCDS, you verify that all classes and groups assigned by your ACS routines are defined in the SCDS. To validate the SCDS:

1. Enter an 8 (CDS) on the ISMF Primary Option Menu to display the CDS Application Selection panel.
2. Enter 4 to Validate the SCDS.

For more information, see *DFSMS/MVS Using the Interactive Storage Management Facility*, SC26-4911.

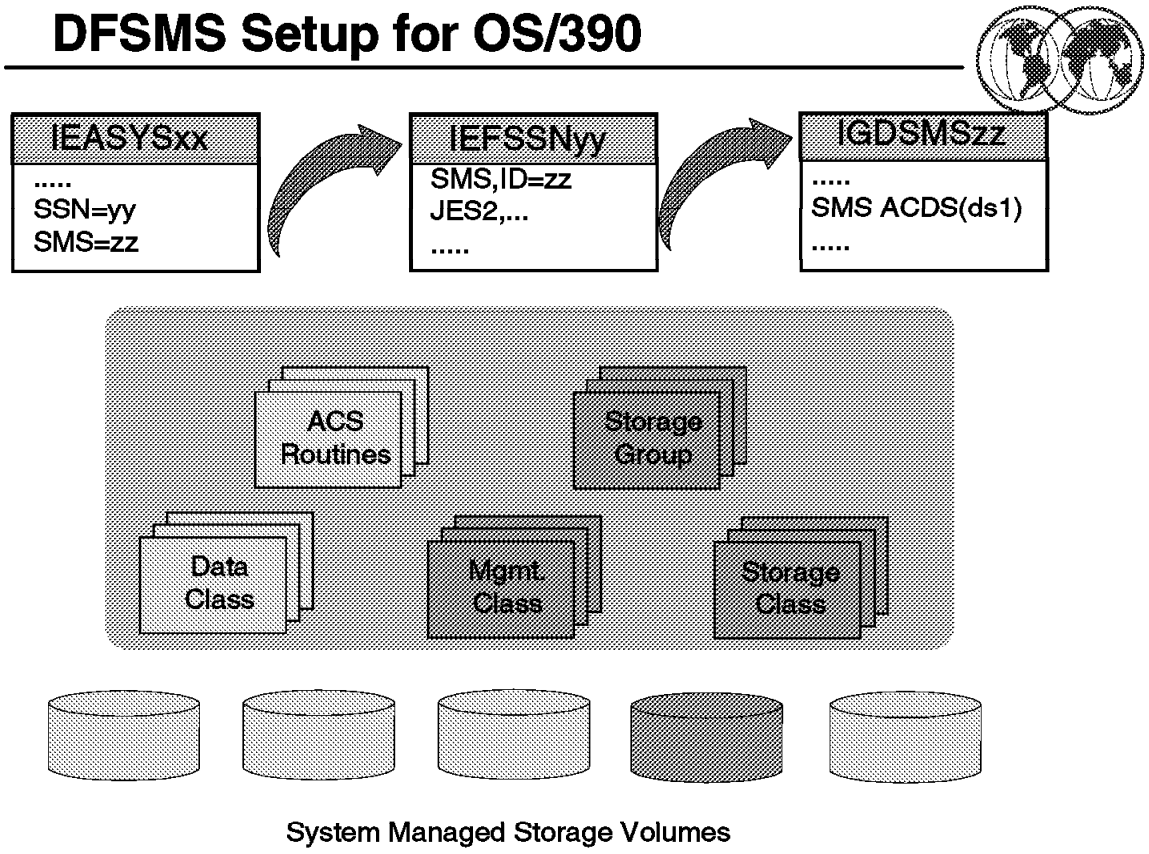


Figure 114. DFSMS setup for OS/390

2.8.6 DFSMS setup for OS/390

In preparation for starting SMS, update the IEASYSxx, IEFSSNyy, and IGDSMSzz members of MVS parmlib to define SMS to MVS.

IEASYSxx. IEASYSxx is used only to specify the suffix of the other two members.

IEFSSNyy. You can activate SMS only after you define it to MVS as a valid subsystem.

To define SMS to MVS, you must place a record for SMS in an IEFSSNxx member. IEFSSNxx defines how MVS activates the SMS address space. You can code an IEFSSNxx member with keyword or positional parameters, but not both. We recommend using keyword parameters. We recommend that you place the SMS record before the JES2 record in IEFSSNxx to start SMS before starting the JES2 subsystem.

IGDSMSzz. For each system in the SMS complex, you must create an IGDSMSxx member SYS1.PARMLIB. The IGDSMSzz member contains SMS initialization control information. It has a default value of 00.

Every SMS system must have an IGDSMSzz member in SYS1.PARMLIB that specifies a required ACDS and COMMDS pair. This ACDS and COMMDS pair is used if the COMMDS of the pair does not point to another COMMDS.

If the COMMDS points to another COMMDS, the referenced COMMDS is used. This referenced COMMDS might contain the name of an ACDS that is different from the one specified in the IGDSMSzz.

If so, the name of the ACDS is obtained from the COMMDS rather than from the IGDSMSzz to ensure that the system is always running under the most recent ACDS and COMMDS.

If the COMMDS of the ACDS and COMMDS pair refers to another COMMDS during IPL, it means a more recent COMMDS has been used. SMS uses the most recent COMMDS to ensure that you cannot IPL with a down-level configuration. The data sets that you specify for the ACDS and COMMDS pair must be the same for every system in an SMS complex. Whenever you change the ACDS or COMMDS, update the IGDSMSzz for every system in the SMS complex so that it specifies the same data sets.

There are much more commands in IGDSMSzz. For a complete description of SMS parameters, see *OS/390 Initialization and Tuning Reference*, SC28-1752, and *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920.

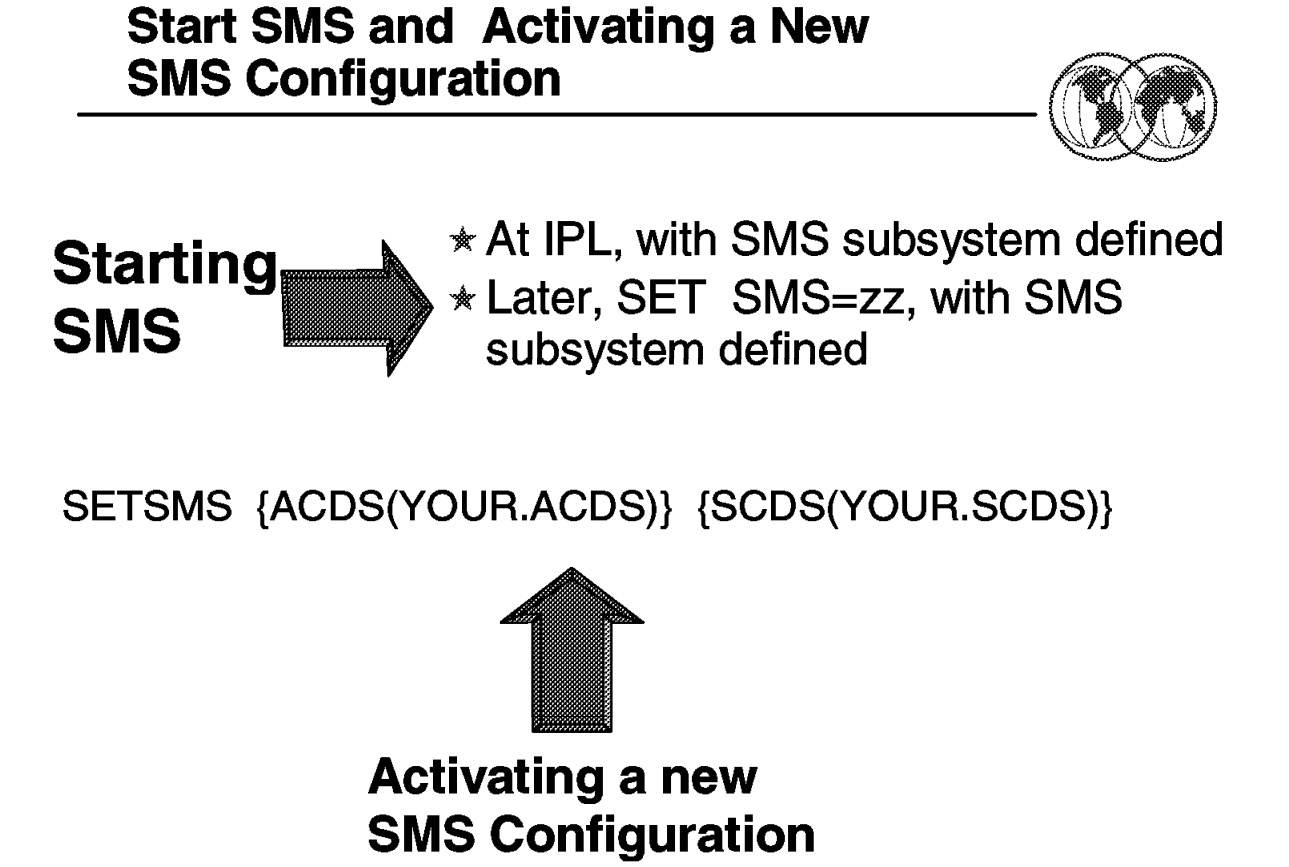


Figure 115. Activating and starting SMS

2.8.7 Activating and starting SMS

To start SMS, which starts the SMS address space, use either of the following two methods:

- IPL the system with SMS defined as a valid subsystem, and start SMS automatically.
- IPL the system with SMS defined as a valid subsystem, and start SMS later using the SET SMS=yy MVS operator command.

For detailed information, refer to the *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920.

2.8.7.1 Activating a new SMS configuration

Activating a new SMS configuration means to copy the configuration from SCDS to ACDS and to SMS address space. You can manually activate a new SMS configuration in two ways. But SMS must be active before you use one of these methods:

- **Activating an SMS configuration from ISMF**

  Select the ACTIVATE option, or enter the ACTIVATE command on the command line of the ISMF CDS Application Selection panel. The information from the SCDS is copied into the ACDS. The SCDS itself is never considered active. Attempting to activate an ACDS that is not valid results in an error message

- **Activating an SMS configuration from the operator console**

  From the operator console, enter the following command:

```
SETSMS {ACDS(YOUR.OWN.ACDS)} {SCDS(YOUR.OWN.SCDS)}
```

Do not confuse the SET SMS (T SMS) command with the SETSMS command.

YOUR.OWN.ACDS specifies a data set that has been defined as an ACDS. To activate the configuration, information is brought into the SMS address space from the ACDS. To update the current ACDS with the contents of an SCDS, specify the name of the SCDS only. Otherwise, If you want to both specify a new ACDS and update it with the contents of an SCDS, enter the SETSMS command with both the ACDS and SCDS specified.

Note: The ACTIVATE command, run from the ISMF CDS application, is equivalent to the SETSMS operator command with the SCDS keyword specified. If you use RACF, you can enable storage administrators to activate SMS configurations from ISMF by defining the facility, STGADMIN.IGD.ACTIVATE.CONFIGURATION, and issuing permit commands for each storage administrator.

Display SMS Configuration

```
 DISPLAY SMS,STORGRP(PRIME80),LISTVOL
00- 20.47.08           D SMS,STORGRP(PRIME80),LISTVOL
    20.47.08           IGD002I 20:47:08 DISPLAY SMS 825
    STORGRP  TYPE              SYSTEM= 1
    PRIME80  POOL                      +
    VOLUME   UNIT    SYSTEM= 1                STORGRP NAME
    PRM801                   +                  PRIME80
    PRM802                   +                  PRIME80
    PRM803                   +                  PRIME80

   ****************************** LEGEND ******************************
   . THE STORAGE GROUP OR VOLUME IS NOT DEFINED TO THE SYSTEM
   + THE STORAGE GROUP OR VOLUME IS ENABLED
   - THE STORAGE GROUP OR VOLUME IS DISABLED
   * THE STORAGE GROUP OR VOLUME IS QUIESCED
   D THE STORAGE GROUP OR VOLUME IS DISABLED FOR NEW ALLOCATIONS ONLY
    Y
   Q THE STORAGE GROUP OR VOLUME IS QUIESCED FOR NEW ALLOCATIONS ONLY
   SYSTEM 1 = SYSTEM1
IEE612I CN=01       DEVNUM=040  SYS=SYSTEM1
```

Figure 116. Display SMS configuration

2.8.8 Display SMS configuration

Display SMS configuration is one of many tasks used to *control SMS processing* as described in 2.8.2, “Steps for a minimal SMS configuration” on page 172.

Display configuration status information by entering ACTIVE in the CDS NAME field on the ISMF CDS Application Selection panel.

MVS operator commands complement your ability to monitor and control SMS operation.

DISPLAY SMS: shows volumes, storage groups, libraries, drives, SMS configuration information, SMS trace parameters, SMS operational options, OAM information, OSMC information, and cache information. Enter this command to:

- Confirm that the system-managed volume status is correct
- Confirm that SMS starts with the proper parameters

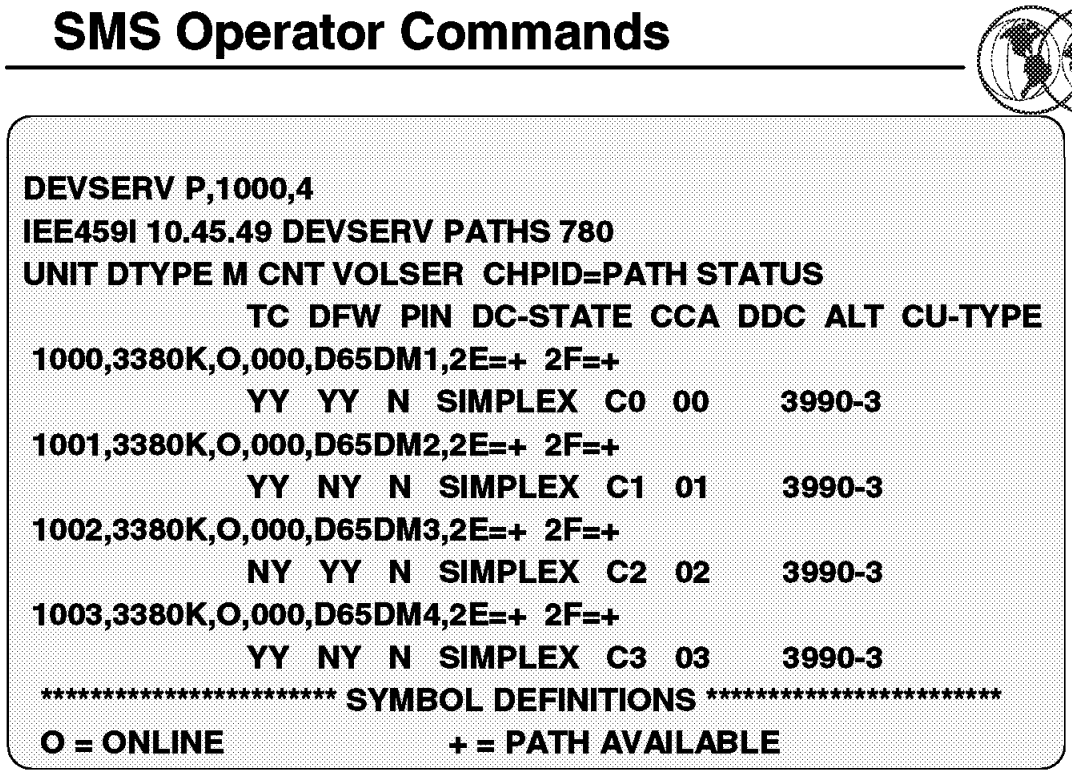


Figure 117. Controlling SMS processing with commands

2.8.9 Controlling SMS processing with MVS operator commands

The DFSMS environment provides a set of MVS commands the operator can use to control SMS processing. The VARY, DISPLAY, DEVSERV, and SET commands are MVS operator commands that support SMS operation.

SETSMS This command changes SMS options from the operator console. You can use this command to activate a new configuration from an SCDS. SETSMS supports SMS and is modeled after the SETSMF command, which controls SMF processing. The MVS operator must SETSMS to recover from ACDS and COMMDS failures. For an explanation of how to recover from ACDS and COMMDS failures, see *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920.

SET SMS=zz This command starts SMS, if it has not already been started and is defined as a valid MVS subsystem. The command also:

- Changes options set from MVS PARMLIB for SMS
- Restarts SMS if it has terminated
- Updates the SMS configuration

VARY SMS This command changes storage group, volume, library, or drive status. You can use this command to:

- Limit new allocations to a volume or storage group
- Enable a newly-installed volume for allocations

DEVSERV This command displays information for a device. Use it to display the status of extended functions in operation for a given volume that is attached to a cache-capable 3990 storage control.

For more information about operator commands, see *OS/390 MVS System Commands*, GC28-1781.

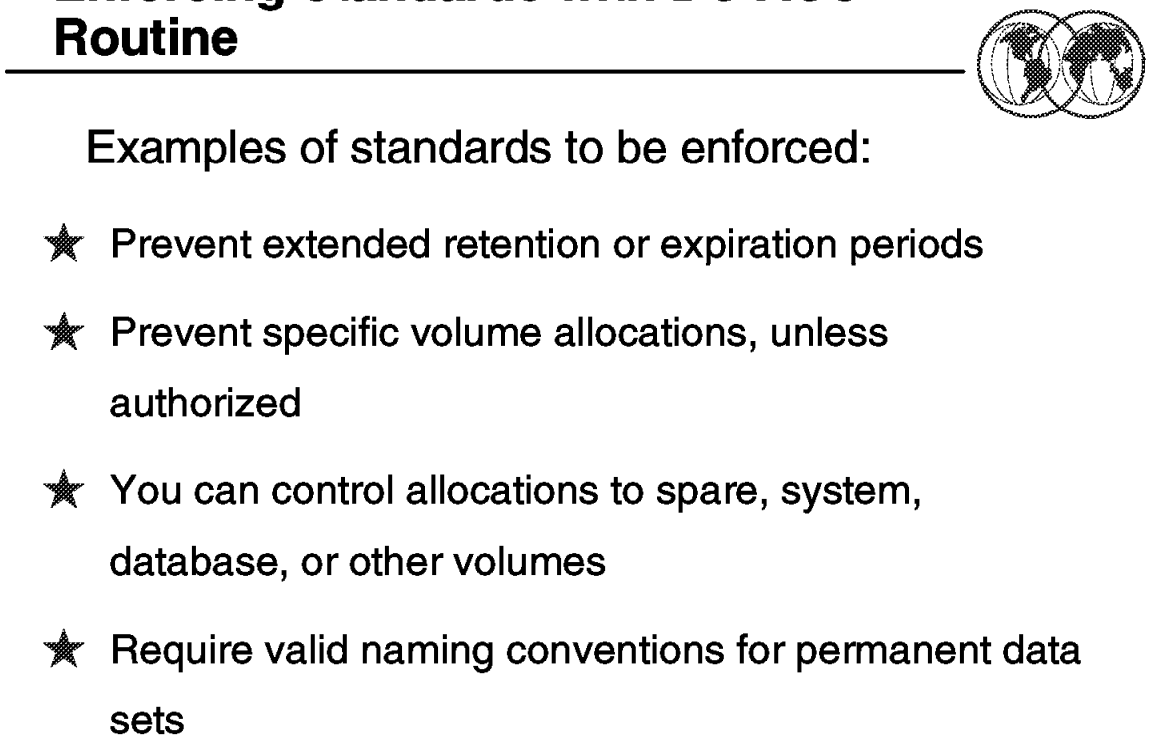


Figure 118. Enforcing standards with DC ACS routine

2.8.10 Enforcing standards with DC ACS routine

You can use data class ACS routine facilities to automate or simplify storage allocation standards if you:

- Use manual techniques to enforce standards
- Plan to enforce standards before implementing DFSMS
- Use DFSMSdfp or MVS installation exits to enforce storage allocation standards

The data class ACS routine provides an automatic method for enforcing standards, because it is called for system-managed and non-system-managed data set allocations. Standards are enforced automatically at allocation time, rather than through manual techniques after allocation.

Enforcing standards optimizes data processing resources, improves service to users, and positions you for implementing system-managed storage. You can fail requests or issue warning messages to users who do not conform to standards. Consider enforcing the following standards in your DFSMS environment:

- Prevent extended retention or expiration periods
- Prevent specific volume allocations, unless authorized; for example, you can control allocations to spare, system, database, or other volumes
- Require valid naming conventions for permanent data sets

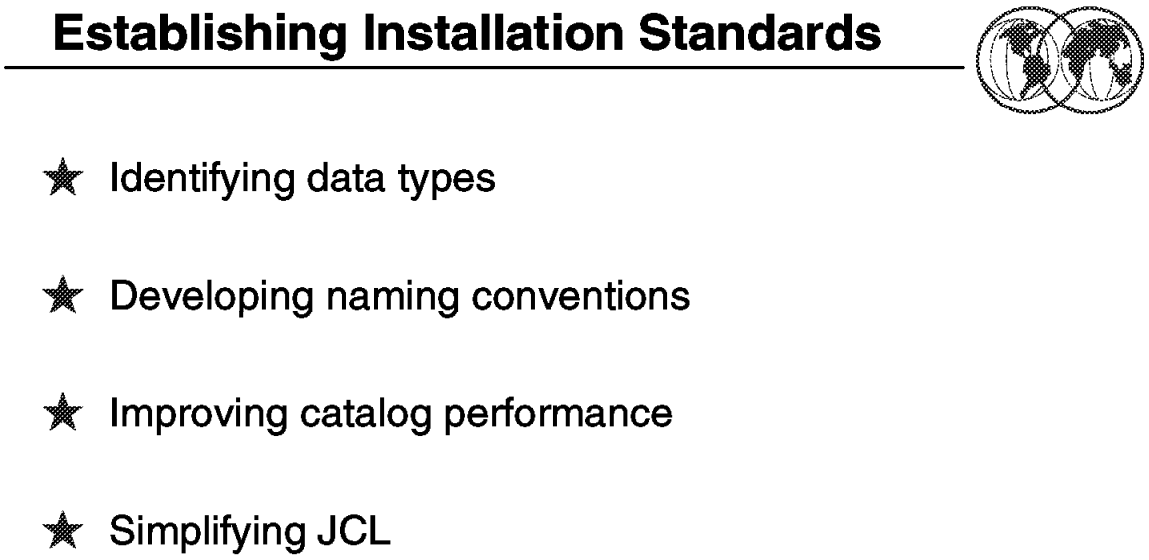


Figure 119. Establishing installation standards

2.9 Establishing installation standards

Establishing standards, such as naming conventions and allocation policies, helps you manage storage more efficiently and improves service to your users. With them, your installation is better prepared to make a smooth transition to system-managed storage. This topic helps you establish installation standards. It describes the following tasks:

- Identifying data types
- Developing naming conventions
- Improving catalog performance
- Simplifying JCL

Negotiate with your user group representatives to agree on the specific policies for the installation, how soon you can implement them, and how strongly you enforce them. Document negotiated policies in a service-level agreement.

You can simplify storage management by limiting the number of data sets and volumes that cannot be system-managed.

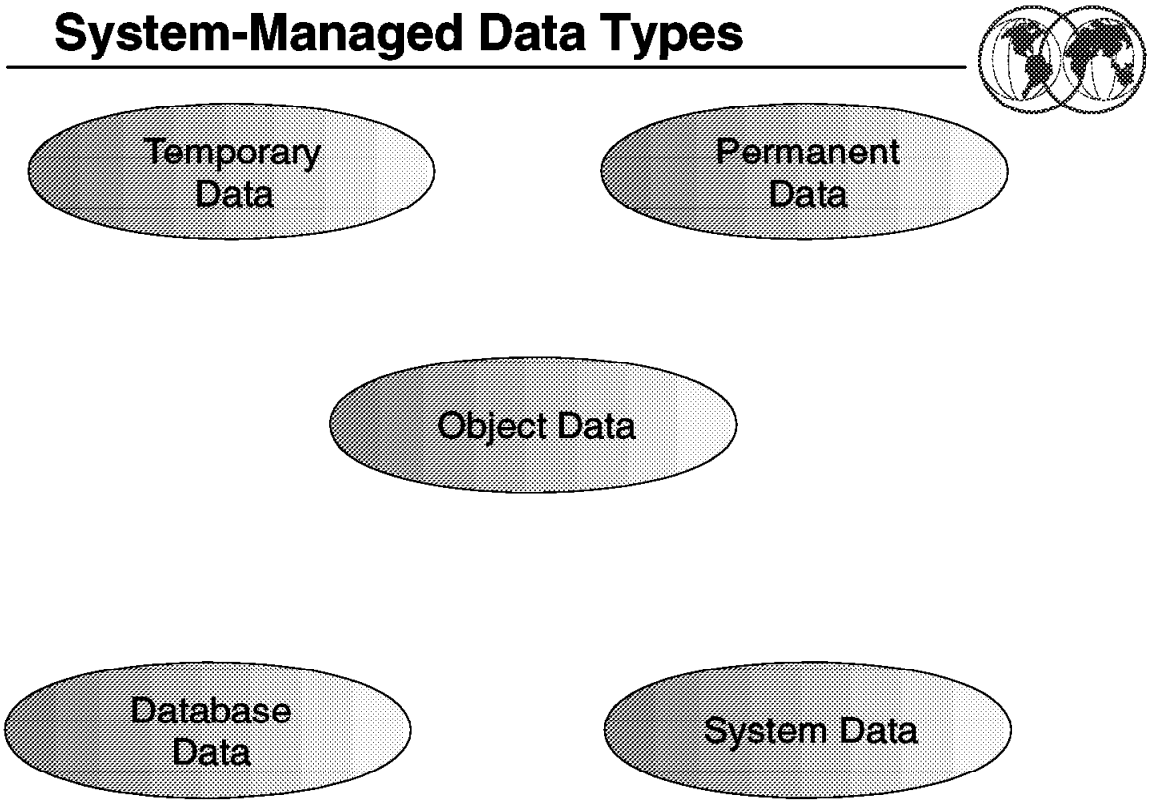


Figure 120. Data types that can be system managed

2.9.1 Data types that can be system managed

These are some common types of data that can be system managed. For details on how these data types can be system managed using SMS storage groups, see *MVS/ESA Storage Management Library Managing Storage Groups*, SC26-3125.

Temporary data Data sets used only for the duration of a job, job step, or terminal session, and then deleted. These data sets can be cataloged or uncataloged, and can range in size from small to very large.

Permanent data Data sets consisting of:

- Interactive data
- TSO user data sets
- ISPF/PDF libraries you use during a terminal session

Data sets classified in this category are typically small, and are frequently accessed and updated.

Batch data Data that is classified as either online-initiated, production, or test.

- Data accessed as online-initiated are background jobs that an online facility, such as TSO, generates
- Production batch refers to data created by specialized applications (such as payroll), that could be critical to the continued operation of your business or enterprise
- Test batch refers to data created for testing purposes

VSAM data Data organized with VSAM, including VSAM data sets that are part of an existing database.

Large data For most installations, large data sets occupy more than 10 percent of a single DASD volume. Note, however, that what constitutes a large data set is installation dependent.

Multivolume data Data sets that span more than one volume.

Database data Data types usually having varied requirements for performance, availability, space, and security. To accommodate special needs, database products have specialized utilities to manage backup, recovery, and space usage. Examples include DB2, IMS, and CICS data.

System data Data used by MVS to keep the operating system running smoothly. In a typical installation, 30-to-50 percent of these data sets are high performance and are used for cataloging, error recording, and other system functions. Because these critical data sets contain information required to find and access other data, they are read and updated frequently, often by more than one system in an installation.

Performance and availability requirements are unique for system data. The performance of the system depends heavily upon the speed at which system data sets can be accessed. If a system data set such as a master catalog is unavailable, the availability of data across the entire system and across other systems can be affected.

Some system data sets can be system-managed if they are uniquely named. These data sets include user catalogs. Place other system data sets on non-system managed volumes. The system data sets which are allocated at MVS system initialization are not system managed, because the SMS address space is not active at initialization time.

Object data Also known as byte-stream data, this data is used in specialized applications such as image processing, scanned correspondence, and seismic measurements. Object data typically has no internal record or field structure and, once written, the data is not changed or updated. However, the data can be referenced many times during its lifetime.

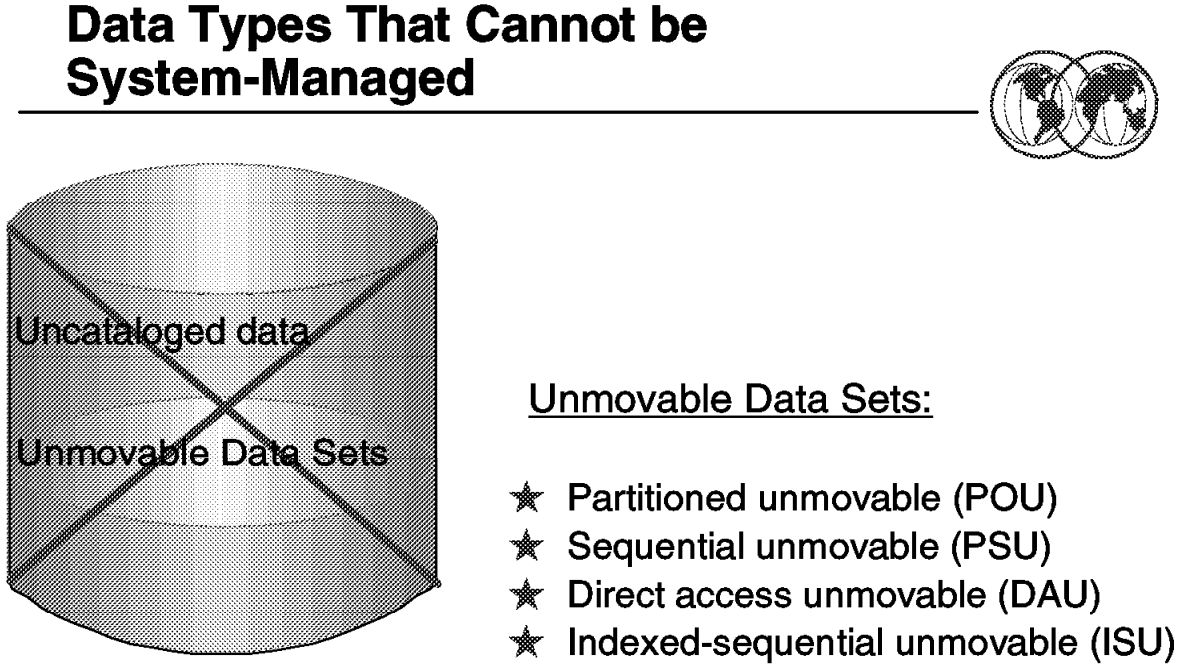


Figure 121. Data types that cannot be system managed

2.9.2 Data types that cannot be system managed

This section describes the data that cannot be system managed.

Uncataloged data

All permanent DASD data under the control of SMS must be cataloged in integrated catalog facility catalogs using the standard search order. The catalogs contain the information required for locating and managing system-managed data.

If data sets are cataloged, users do not need to know which volumes the data sets reside on when they reference them; they do not need to specify unit type or volume serial number. This is essential in an environment with storage groups, where users do not have private volumes.

Objects, stored in groups called collections, must have their collections cataloged in integrated catalog facility catalogs because they, and the objects they contain, are system-managed data. The object access method (OAM) identifies an object by its collection name and the object's own name. An object is described only by an entry in a DB2 object directory. An object collection is described by a collection name catalog entry and a corresponding OAM collection identifier table entry. Therefore, an object is accessed by using the object's collection name and the catalog entry.

When objects are written to tape, they are treated as tape data sets and OAM assigns two tape data set names to the objects. Objects in an object storage group being written to tape are stored as a tape data set named

OAM.PRIMARY.DATA. Objects in an object backup storage group being written to tape are stored as a tape data set named OAM.BACKUP.DATA. Each tape containing objects has only one tape data set, and that data set has one of the two previous names. Because the same data set name can be used on multiple object-containing tape volumes, the object tape data sets are not cataloged.

If you don't already have a policy for cataloging all permanent data, it is a good idea to establish one now. For example, you can enforce standards by deleting uncataloged data sets.

Uncataloged data sets The system locates these with JOBCAT or STEPCAT statements. Data set LOCATEs using JOBCATs or STEPCATs are not permitted for system-managed data sets. You must identify the owning catalogs before you migrate these data sets to system management. The ISPF/PDF SUPERC utility is valuable for scanning your JCL and identifying any dependencies on JOBCATs or STEPCATs.

Unmovable data sets Unmovable data sets cannot be system-managed. These data sets include:

- Data sets identified by the following data set organization (DSORGs):
  - Partitioned unmovable (POU)
  - Sequential unmovable (PSU)
  - Direct access unmovable (DAU)
  - Indexed-sequential unmovable (ISU)
- Data sets with user-written access methods
- Data sets containing processing control information on the device or volume on which they reside, including:
  - Absolute track data that is allocated in absolute DASD tracks or on split cylinders
  - Location dependent direct data sets

All unmovable data sets must be identified and converted for use in a system-managed environment. For information on identifying and converting unmovable data sets, see *MVS/ESA Storage Management Library Managing Storage Groups*, SC26-3125.

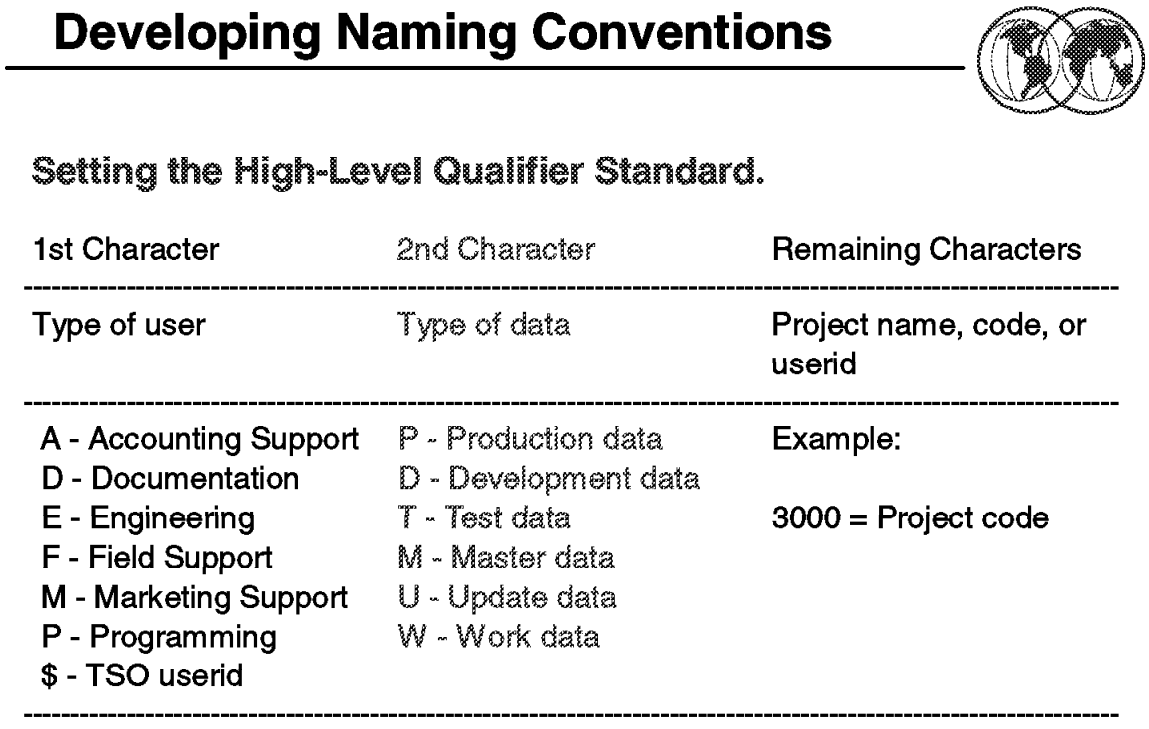


Figure 122. Highest-level qualifiers

2.9.3 Developing naming conventions

Whenever you allocate a new data set, you (or the operating system) must give the data set a unique name. Usually, the data set name is given as the dsname in JCL. A data set name can be one name segment, or a series of joined name segments. Each name segment represents a level of qualification. For example, the data set name DEPT58.SMITH.DATA3 is composed of three name segments. The first name on the left is called the high-level qualifier, the last is the low-level qualifier.

You must implement a naming convention for your data sets. Although naming convention is not a prerequisite for DFSMS conversion, it makes more efficient use of DFSMS. You can also reduce the cost of storage management significantly by grouping data that shares common management requirements. Naming conventions are an effective way of grouping data. They also:

- Simplify service-level assignments to data
- Facilitate writing and maintaining ACS routines
- Allow data to be mixed in a system-managed environment while retaining separate management criteria
- Provide a filtering technique useful with many storage management products
- Simplify the data definition step of aggregate backup and recovery support

Naming conventions are particularly important to data in a distributed environment. For more information, see *ADSTAR Distributed Storage Manager for MVS Administrator's Guide*, GC35-0277.

This section explains the highest-level and lowest-level qualifiers (HLQ and LLQ, respectively), and other levels. Most naming conventions are based on the HLQ and LLQ of the data name.

Other levels of qualifiers can be used to identify generation data sets and database data. They can also be used to help users to identify their own data.

2.9.3.1 Highest-level qualifiers (HLQ)

The HLQ identifies the owner or owning group of the data, and it can indicate data type. Do not embed information that is subject to frequent change in the HLQ, such as department number, application location, output device type, job name, or access method. Set a standard within the HLQ. This visual shows a suggestion.

Setting the Low-Level Qualifier Standard

Low-Level Qualifier	Exp Days Non-Usage	Max Ret Period	Partial Release	Migrate Days Non-Usage	Cmd/Auto Migrate	No.GDG Primary	Backup Freqcy	Backup Versns	Retain Days Only BUP	Retain Day Extra BUP
ASM.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
CLIST....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
COB*.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
CNTL.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
DATA.....	400	400	YES	15	BOTH	--	2	2	400	60
*DATA....	400	400	YES	15	BOTH	--	2	2	400	60
FOR*.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
INCL*....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
INPUT....	400	400	YES	15	BOTH	--	2	2	1100	120
ISPROF...	400	400	YES	30	BOTH	--	0	2	60	30
JCL.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
LIST*....	2	2	YES	NONE	NONE	--	NONE	NONE	--	--
*LIST....	2	2	YES	NONE	NONE	--	NONE	NONE	--	--
LOAD*....	400	400	YES	15	BOTH	--	1	2	--	--
MACLIB...	400	400	YES	15	BOTH	--	1	2	400	60
MISC.....	400	400	YES	15	BOTH	--	2	2	400	60
NAMES...	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
OBJ*.....	180	180	YES	7	BOTH	--	3	1	180	30
PLI.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120

Figure 123. Lowest-level qualifiers (LLQ) standards

2.9.4 Lowest-level qualifiers (LLQ) standards

The LLQ determines the contents and storage management processing of the data. You can use LLQs to identify data requirements for:

- Migration (data sets only)
- Backup (data sets and objects)
- Archiving (data sets)
- Retention or expiration (data sets and objects)
- Class transitions (objects only)
- Release of unused space (data sets only)

The retention and expiration of objects on tape volumes are determined on two levels. Tape volumes containing objects have a tape data set expiration date and an expiration date of when the last object on the tape is going to expire. For information on deleting expired objects on tape, see “Deleting Data” in topic 4.2 in *MVS/ESA Storage Management Library Managing Data*, SC26-3124.

Mapping storage management requirements to data names is especially useful in a system-managed environment. In an environment without storage groups, data with differing requirements is often segregated onto separate volumes that are monitored and managed manually. LLQ data naming conventions allow data to be mixed together in a system-managed environment and still retain the separate management criteria.

This visual shows some examples of how you can use LLQ naming standards to indicate the storage management processing criteria. The first column lists the LLQ of a data name. An asterisk indicates

where a partial qualifier can be used. For example, LIST* indicates that only the first four characters of the LLQ must be LIST; valid qualifiers include LIST1, LISTING, and LISTOUT. The remaining columns show the storage management processing information for the data listed.

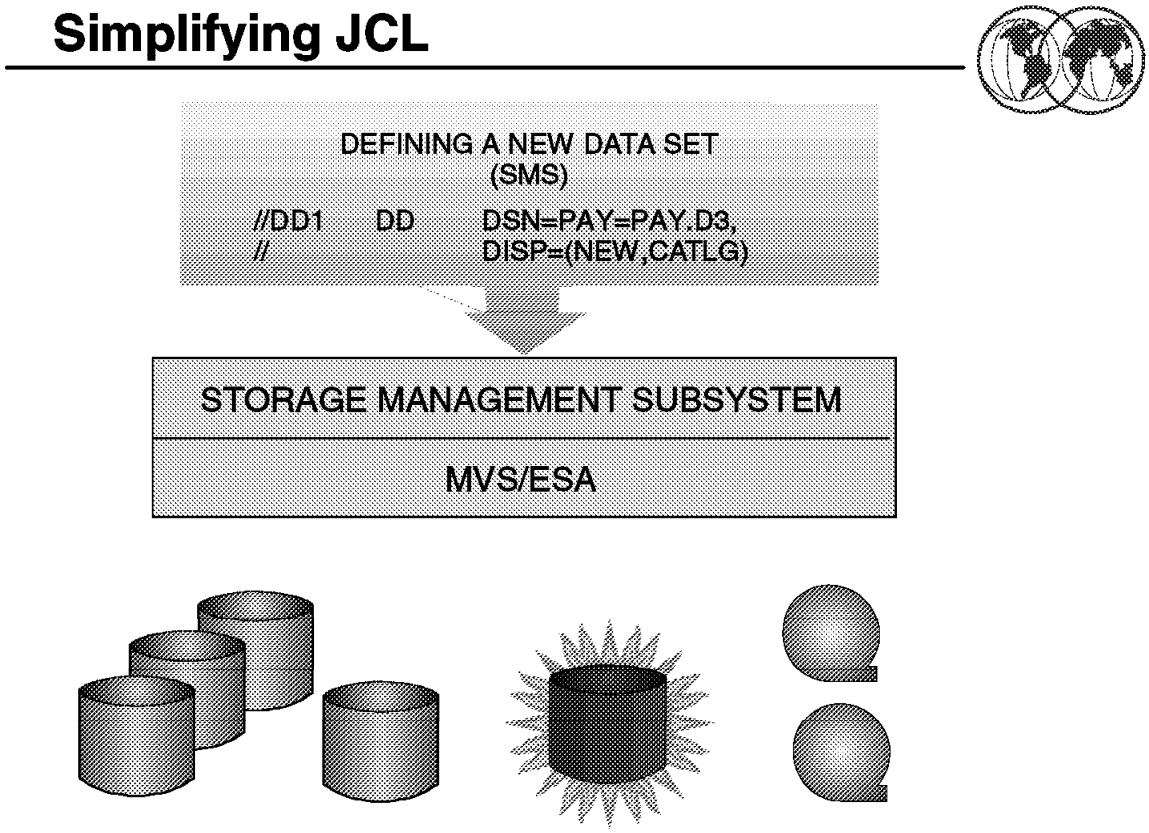


Figure 124. Simplifying JCL

2.9.5 Simplifying JCL

Several JCL keywords can help you simplify the task of creating data sets and also to make the allocation process under TSO more consistent. It is also possible to allocate VSAM data sets through JCL without IDCAMS assistance.

For example, with the use of data classes, you have less use for the JCL keywords: VOL, UNIT, DCB, and AMP. Continue to use the DSN, DISP, and SPACE keywords to create system-managed data.

2.9.5.1 JCL keywords used in the DFSMS environment

You can use these JCL keywords to create VSAM and non-VSAM data sets. For a detailed description of the keywords and their use, see *OS/390 MVS JCL User's Guide*, SC28-1758.

2.9.5.2 Sample jobs using the JCL keywords

This section gives examples of how to use JCL keywords when:

- Creating a data set
- Creating a VSAM cluster
- Specifying retention period
- Specifying expiration date

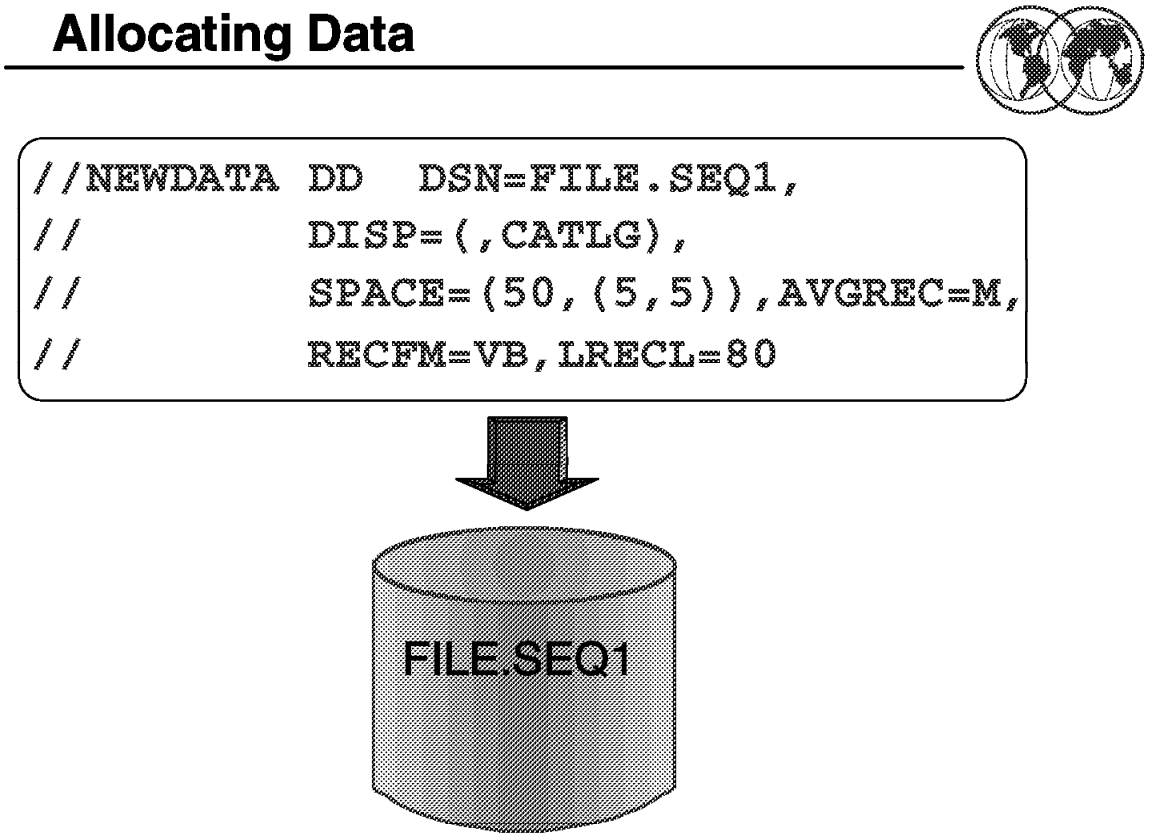


Figure 125. Allocating data

2.9.6 Allocating data

Many times the words *create* and *allocate*, when applied to data sets, are used in MVS as synonyms. However, they are not.

- To *create* (in DASD) means to assign a space in VTOC to be used for a data set (sometimes *create* implies cataloging the data set). A data set is created in response to the DD card DISP=NEW in JCL.
- To *allocate* means to establish a logical relationship between the request for the use of the data set within the program (through the use of an DCB or ACB) and the data set itself in the device where it is located. Being more specific, allocation implies finding where the data set is (for an already existent data set) or where it will be (for a new one). Thinking in control blocks terms, the DCB/ACB is connected to the DD card through the DDNAME field. The DD card contents forms a TIOT entry and at allocation time this entry points to the UCB where the data set exists.

This visual shows an example of JCL used to create a data set in a system-managed environment.

These are some characteristics of the JCL in a system-managed environment:

- The device-dependent volume serial number and unit information is no longer required, because the volume is assigned within a storage group selected by the ACS routines. This eliminates the need for device awareness.

- The LRECL and RECFM parameters are independent keywords. This makes it easier to override individual attributes that are assigned default values by the data class, selected by the ACS routines, that might not be appropriate for the data set being allocated.
- In this example, the SPACE parameter is coded with the average number of bytes per record (50), and the number of records required for the primary data set allocation (5 M) and secondary data set allocation (5 M). These are the values that the system uses to calculate the least number of tracks required for the space allocation. This also eliminates the need for device awareness, replacing the TRK or CYL unit specification. For variable-block data sets, the average number of bytes per record is not necessarily the same as the LRECL value. In the example, the average record length is 50, whereas the LRECL is 80.

  The AVGREC attribute indicates the scale factor for the primary and secondary allocation values. In the example, an AVGREC value of M indicates that the primary and secondary values of 5 are each to be multiplied by 1 048 576.

  The SPACE parameter would result in a primary allocation of 5 MB and a secondary allocation of 5 MB.

  The JCL to allocate a data set under SMS is simpler and has no device-dependent keywords. For information on managing data allocation, refer to *DFSMS/MVS Using Data Sets*, SC26-4922.

  Note that, if you code the SPACE parameter on a DD statement that defines an existing data set, the SPACE value you specify temporarily overrides the SPACE value used to create the data set.

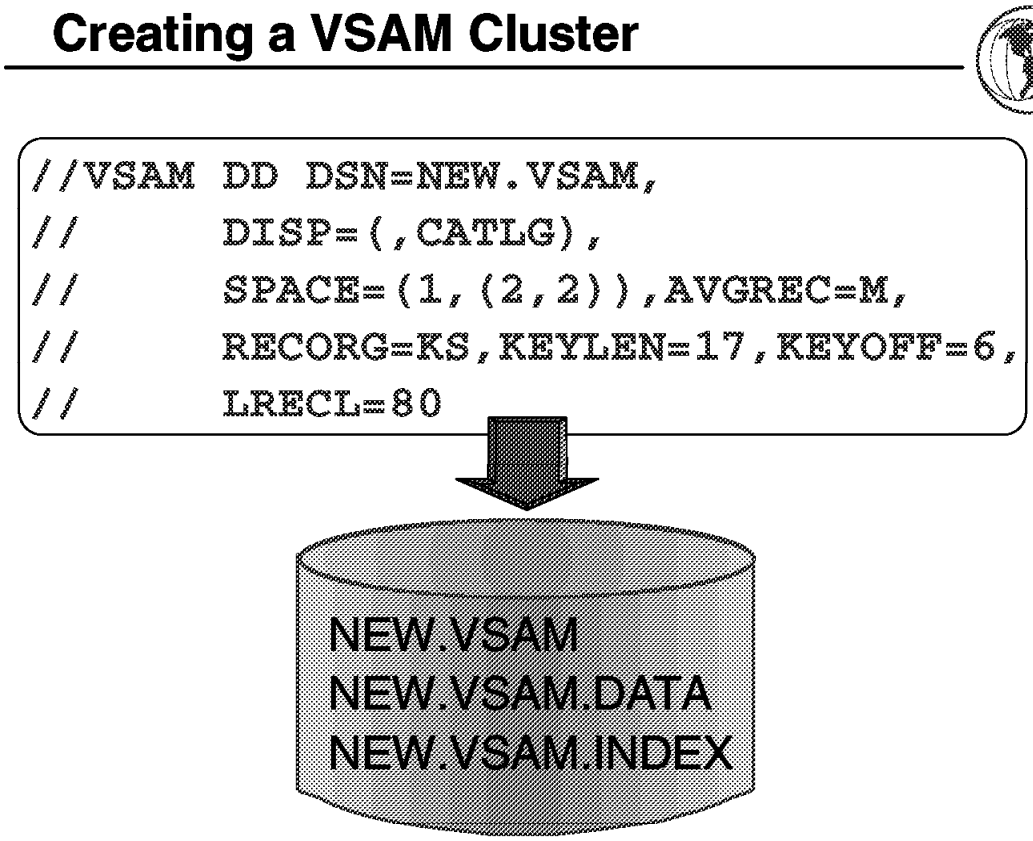


Figure 126. Creating a VSAM cluster

2.9.7 Creating a VSAM cluster

In the DFSMS environment, you can allocate VSAM data sets using JCL or data class. For a non-DFSMS VSAM cluster, this is only done through IDCAMS. In JCL, DISP=(OLD,DELETE) is not ignored for VSAM data sets, and causes the VSAM data set to be deleted at unallocation. A data set with a disposition of MOD is treated as a NEW allocation if it does not already exist; otherwise, it is treated as an OLD allocation.

Space Parameter in KSDS VSAM Cluster

- Allocation specified at cluster or alternate index level
- Allocation specified at the data level
- Allocation specified at both the data and index levels
- Secondary allocation specified at the data level

Figure 127. Space parameter in a KSDS VSAM cluster

2.9.7.1 Space parameter in VSAM KSDS cluster

In this example DSN=NEW.VSAM refers to a KSDS VSAM cluster.

The space allocation for a VSAM entity depends on the level of the entity being allocated:

- If allocation is specified at the cluster or alternate index level only, the amount needed for the index is subtracted from the specified amount. The remainder of the specified amount is assigned to data.
- If allocation is specified at the data-level only, the specified amount is assigned to data. The amount needed for the index is in addition to the specified amount.
- If allocation is specified at both the data and index levels, the specified data amount is assigned to data and the specified index amount is assigned to the index.
- If secondary allocation is specified at the data level, secondary allocation must be specified at the index level or the cluster level.

You cannot use certain parameters in JCL when allocating VSAM data sets, although you can use them in the IDCAMS `DEFINE` command.

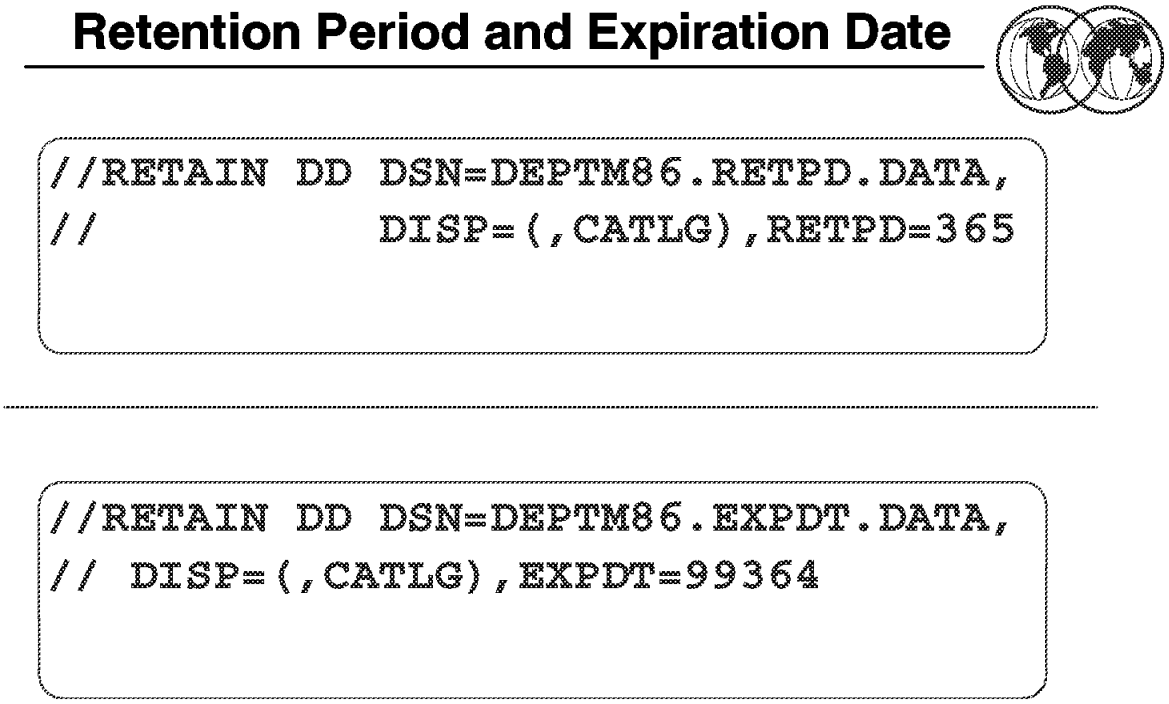


Figure 128. Using retention period and expiration date

2.9.8 Using retention period and expiration date

Parameters for specifying retention period and expiration date control the time during which a data set is protected from being deleted by the system. The first DD statement in the visual protects the data set from deletion for 365 days. The second DD statement in the picture protects the data set from deletion until December 30, 1999.

The RETPD and EXPDT keywords apply alike to system-managed and non-system-managed data sets.

The VTOC entry for both non-VSAM and VSAM data sets contains the expiration date as declared in the JCL, the TSO ALLOCATE command, or the IDCAMS DEFINE command. The expiration date can also come from the data class definition.

The expiration date is placed in the VTOC either directly from the date specification, or after it is calculated from the retention period specification. The expiration date in the catalog entry exists for information purposes only.

If you specify the current date or an earlier date, the data set is immediately eligible for replacement.

You can use management class to limit or ignore RETPD and EXPDT parameters given by a user. If a user specifies values that exceed the maximum allowed by the management class definition, the retention period is reset to the allowed maximum. For more information on using management class to control retention period and expiration date, refer to *DFSMShsm Storage Administration Guide*, SH21-1076.

For an expiration date beyond year 1999 use the following format: YYYY/DDD.

Note: EXPDT=99365, or 99366, or 1999/365 or 1999/3666 are special dates and mean *never expires*.

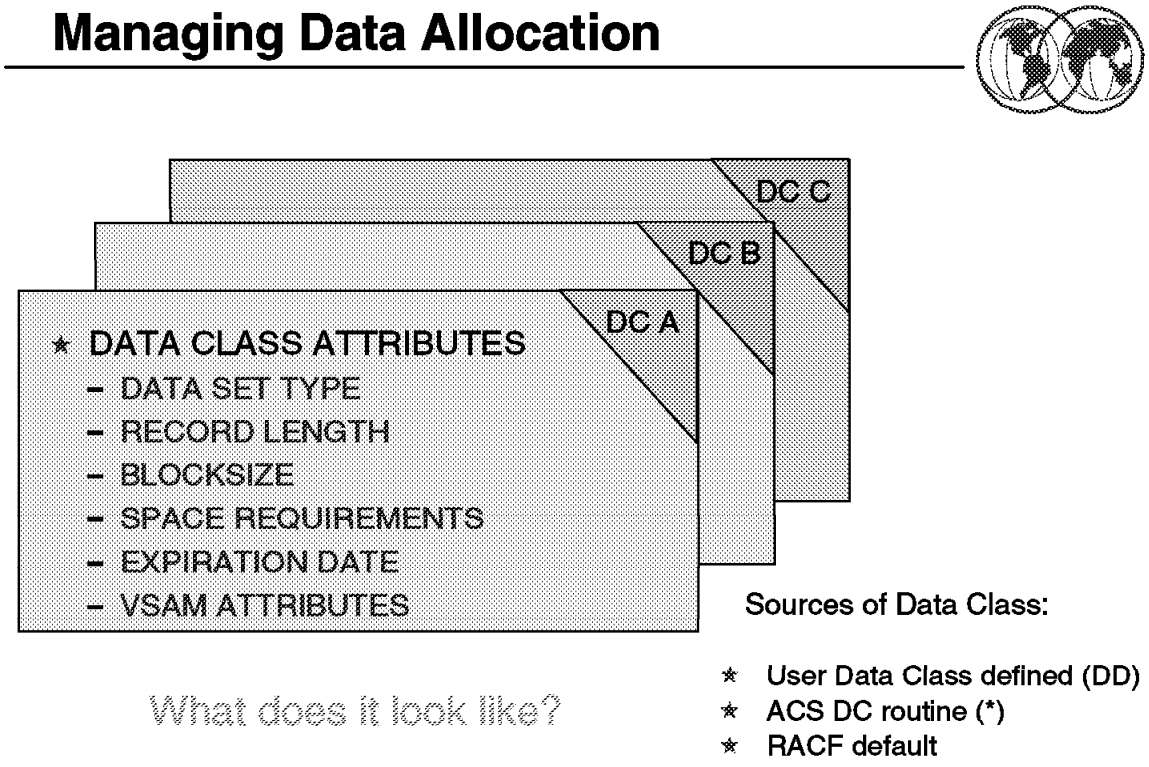


Figure 129. Managing data allocation

2.10 Managing data allocation

Inefficient space usage and poor data allocation cause problems with space and performance management. In a DFSMS environment, you can enforce good allocation practices to help reduce some of these problems. This section describes how to do this by:

- Establishing allocation standards
- Ensuring device independence
- Allocating data sets as PDSEs
- Processing GDGs
- Allocating compressed data

2.10.1 Using data class to standardize data allocation

This section describes how to use data class to establish standards for data allocation. For sample data classes, descriptions, and ACS routines, see *MVS/ESA Implementing System-Managed Storage*, SC26-3123.

You can define data classes containing standard data set allocation attributes. Users then only need to use the appropriate data class names to create standardized data sets. To override values in the data class definition, they can still provide specific allocation parameters.

Data classes can be determined from the user-specified value on the DATACLAS parameter (DD card, TSO Alloc, Dynalloc macro), from a RACF default, or by ACS routines. ACS routines can also override user-specified or RACF default data classes. The asterisk in the visual is highlighting this fact.

However, you can override a data class attribute (not the data class itself) using JCL or dynamic allocation parameters. However, overriding a subparameter of a parameter, overrides all of the subparameters for that parameter. For example, SPACE=(TRK,(1)) in JCL will cause primary, secondary, and directory quantities, as well as AVGREC and AVGUNIT, in the data class to be overridden. DFSMS usually does not change values that are explicitly specified because doing so would alter the original meaning and intent of the allocation. There is an exception—if it is clear that a PDS is being allocated (DSORG=PO or DSNTYPE=PDS is specified), and no directory space is indicated in the JCL, then the directory space from the data class is used even though SPACE=(TRK,(1)) was specified.

Users cannot override the data class attributes of dynamically-allocated data sets if you use the IEFDB401 user exit.

Data classes can also be determined for objects by a specification using the SETOAM command in the CBROAMxx member of SYS1.PARMLIB.

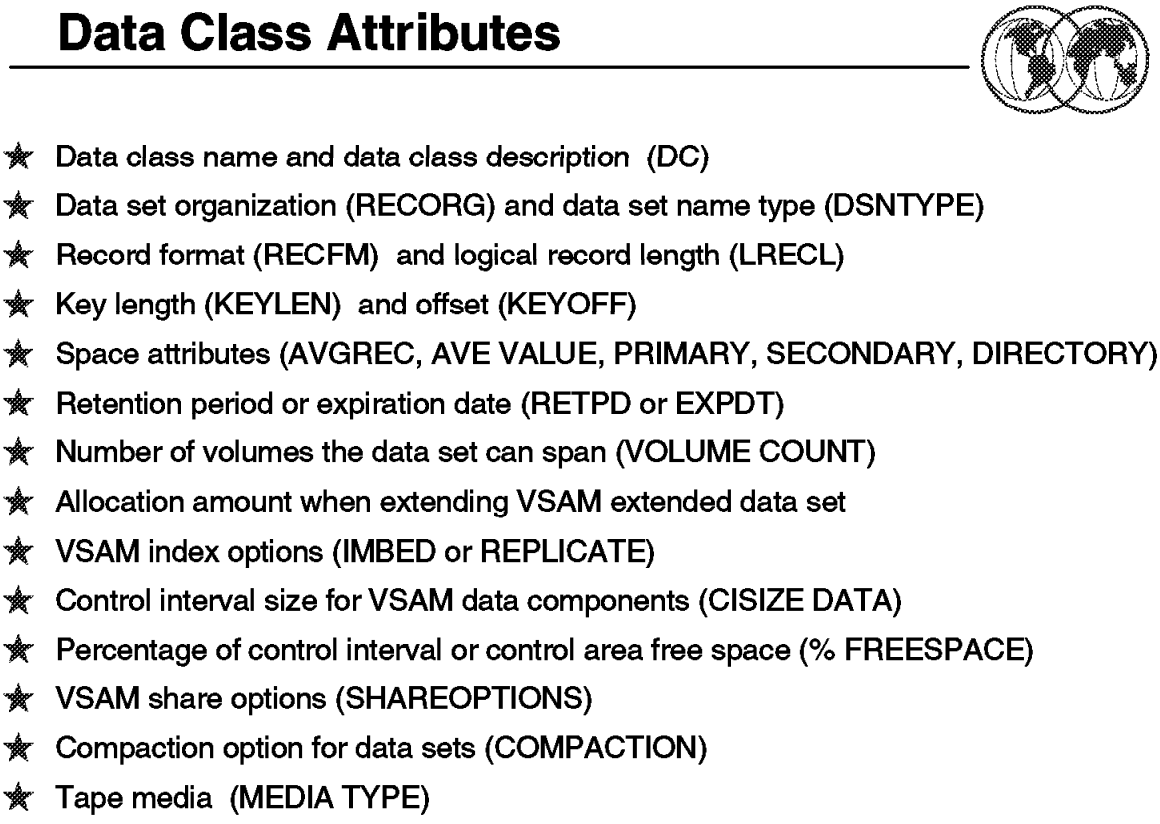


Figure 130. Data class attributes

2.10.2 Data class attributes

The data class attributes you can use are shown in this visual. You can specify the data-class space attributes to control DASD space waste. For example, the primary space value should specify the total amount of space initially required for output processing. The secondary allocation allows automatic extension of additional space as the data set grows and does not waste space by over-allocating the primary quantity. You can also use the data-class space attributes to relieve users of the burden of calculating how much primary and secondary space to allocate.

The COMPACTION attribute specifies whether data is to be compressed on DASD if the data set is allocated in the extended format. The COMPACTION attribute alone also allows you to use the improved data recording capability (IDRC) of your tape device when allocating tape data sets.

The MEDIA TYPE and RECORDING TECHNOLOGY attributes are used for tape data sets only. MEDIA TYPE allows you to select the mountable tape media cartridge type. RECORDING TECHNOLOGY allows you to select the format to use when writing to that device.

The read-compatible special attribute indicator in the tape device selection information (TDSI) allows an 18 track tape to be mounted on a 36 track device for read access. The attribute increases the number of devices that are eligible for allocation when you are certain that no more data will be written to the tape.

For detailed information on specifying data class attributes, see *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920.

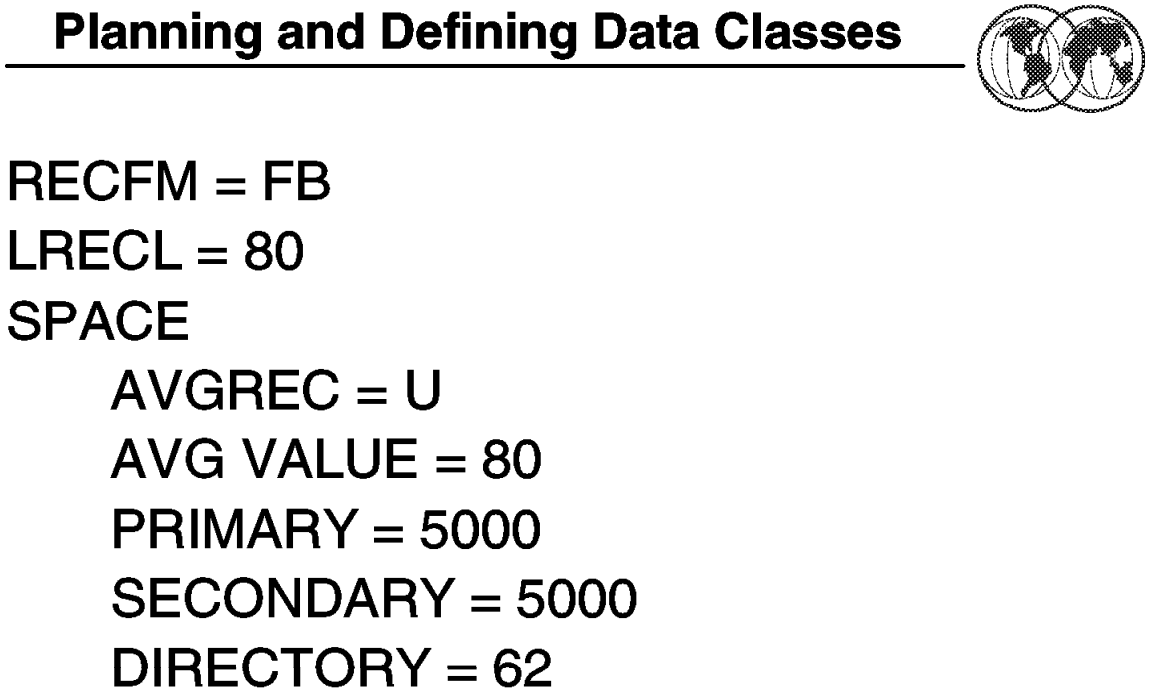


Figure 131. Planning and defining data classes

2.10.3 Planning and defining data classes

Use your service-level agreement (SLA) for reference when you plan your data classes. For information about SLAs, see *MVS/ESA Storage Management Library Leading a Storage Administration Group*, SC26-3126. SLAs identify users′ current allocation practices and their requirements. For example, based on user requirements, you might create a data class to allocate standard control libraries in CDS data sets.

If you want the data class to supply the default value of a parameter, do not specify a value for that parameter in the JCL or dynamic allocation.

Data class names should indicate the type of data they are assigned to. This makes it easier for users to identify the template they need to use for allocation.

You define data classes using the ISMF data class application. Users can access the Data Class List panel to determine which data classes are available and the allocation values that each data class contains.

For more information on planning and defining data classes, see *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920.

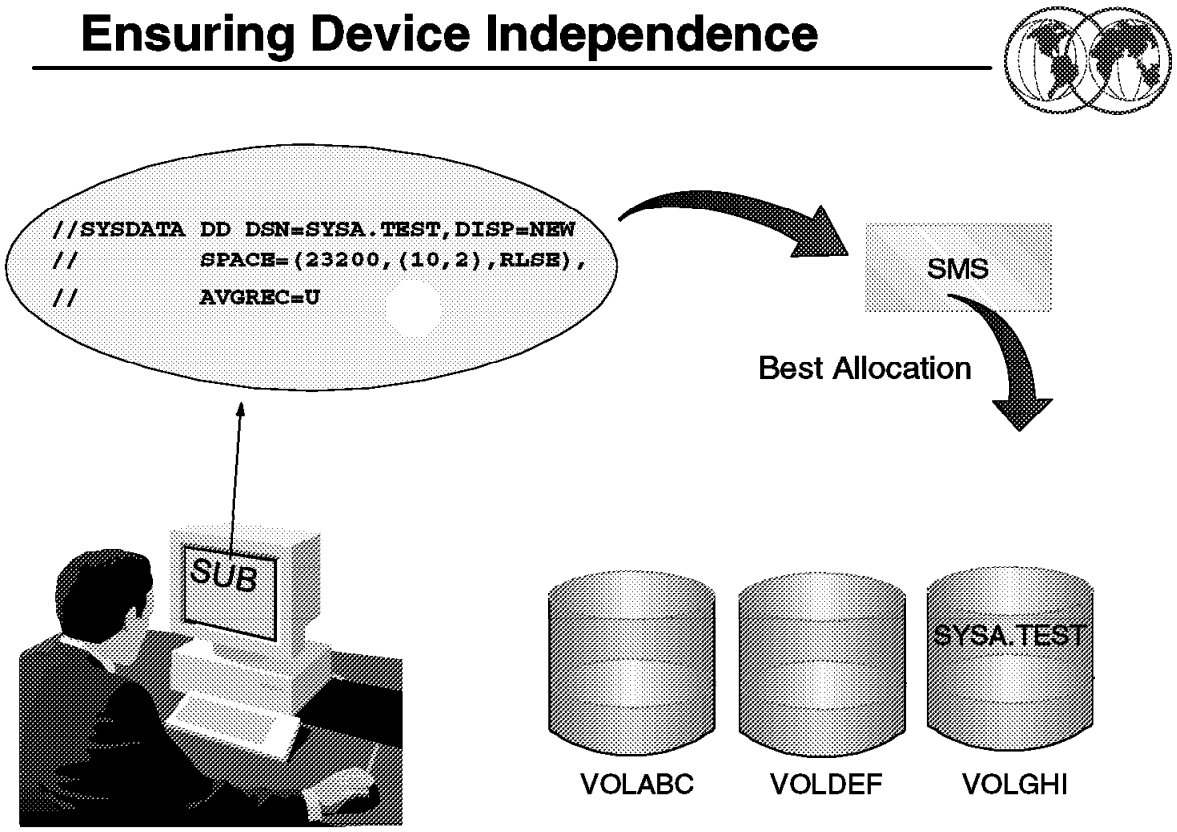


Figure 132. Ensuring device independence

2.10.4 Ensuring device independence

To take full advantage of system-managed storage and improve space usage, you need to eliminate device dependencies. This allows the system to place data on the most appropriate device in the most efficient way. This visual shows how to ensure device independence by removing volume serial number dependencies.

In the DFSMS environment, you control volume selection through the storage class and storage group definitions you create, and by ACS routines. This means that users do not have to specify volume serial numbers with the VOL=SER parameter, or code a specific device type with the UNIT= parameter on their JCL. When converting data sets for use in DFSMS, they do not have to remove these parameters from existing JCL because volume and unit information can be ignored with ACS routines. However, work with users to evaluate UNIT and VOL=SER dependencies before conversion.

Note: If you keep the VOL=SER parameter for a non-SMS volume, but you are trying to access a system-managed data set, then SMS might not find the data set. All SMS data sets (the ones with a storage class) must reside in a system-managed volume.

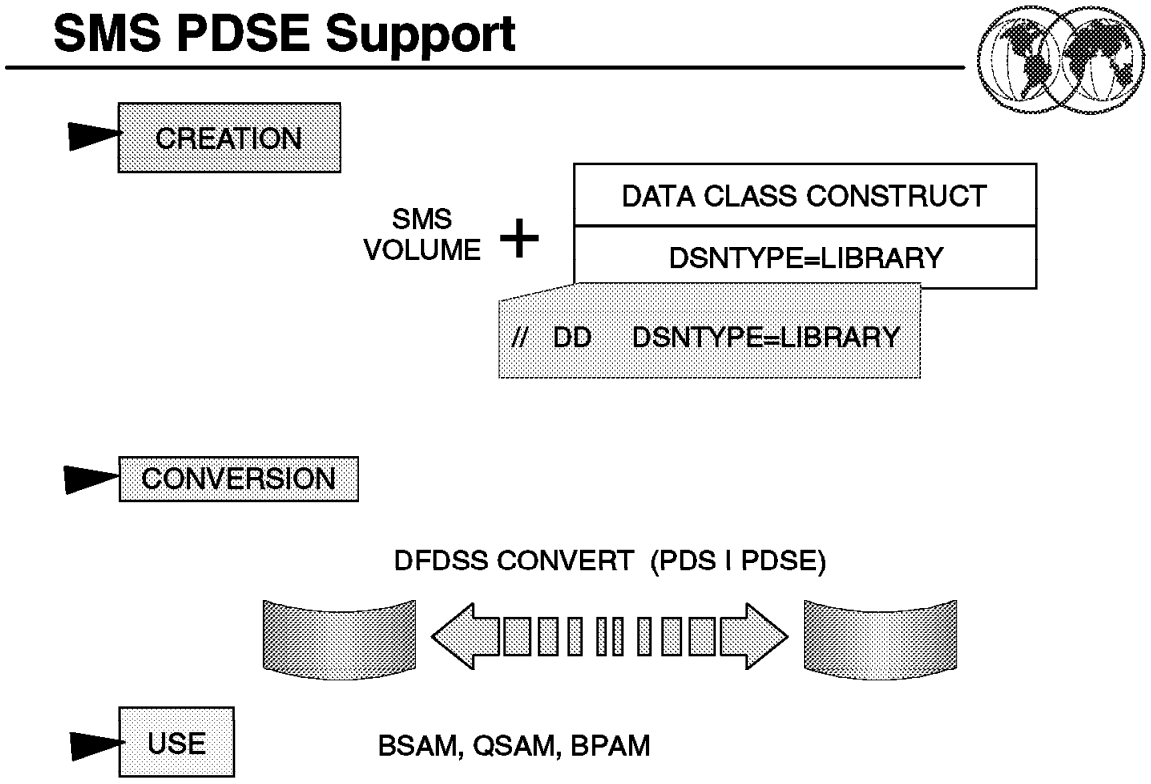


Figure 133. SMS PDSE support

2.11 SMS PDSE support

Partitioned data set extended (PDSE) is a type of data set organization which improves the partition data set (PDS) organization. It has an improved indexed directory structure and a different member format. All PDSE data sets must be system-managed. You can use them for source (programs and text) libraries, macros, and program object libraries. PDSE advantages, when compared with PDS are:

- The size of a PDSE directory is flexible and can expand to accommodate the number of members stored in it (the size of a PDS directory is fixed at allocation time).
- PDSE members are indexed in the directory by member name. This eliminates the need for time-consuming sequential directory searches.
- The logical requirements of the data stored in a PDSE are separated from the physical (storage) requirements of that data, which simplifies data set allocation.
- PDSEs provide more efficient use of DASD space. For example, by moving or deleting a PDSE member, you free space that is immediately available for the allocation of a new member, without first having to compress the data set to consolidate the fragmented space for reuse. This makes PDSEs less susceptible to space-related abends than PDSs.
- The number of PDSE members stored in the library can be large or small without concern for performance or space considerations.
- The ability to update a member in place is possible with PDSs and PDSEs. But with PDSEs, you can extend the size of members and the integrity of the library is maintained while simultaneous changes are made to separate members within the library.

- The maximum number of extents of a PDSE is 123; the PDS is limited to 16.
- PDSEs are device-independent because they do not contain information that depends on location or device geometry.
- All members of a PDSE are reblockable.
- PDSEs can contain program objects built by the program management binder that cannot be stored in PDSs.

You can also share PDSEs within and across systems. With systems that support PDSEs (MVS/DFP 3.2.0 or higher level), multiple users are allowed to read PDSE members while the data set is open.

If you have DFSMS/MVS installed, you can extend the sharing to enable multiple users on multiple systems to concurrently create new PDSE members and read existing members.

Using the PDSESHARING keyword in the SYS1.PARMLIB member, IGDSMSxx, you can specify:

- *NORMAL*. Allows multiple users to read any member of a PDSE
- *EXTENDED*. Allows multiple users to read any member or create new members of a PDSE.

All systems sharing PDSEs need to be upgraded to DFSMS/MVS to use the extended PDSE sharing capability.

After updating the IGDSMSxx member of SYS1.PARMLIB, you need to issue the `SET SMS ID=xx` command for every system in the complex to activate the sharing capability.

For additional information on PDSEs, see *DFSMS/MVS Using Data Sets*, SC26-4922.

Although SMS supports PDSs, you should consider converting these to the PDSE format. The following sections describe this process.

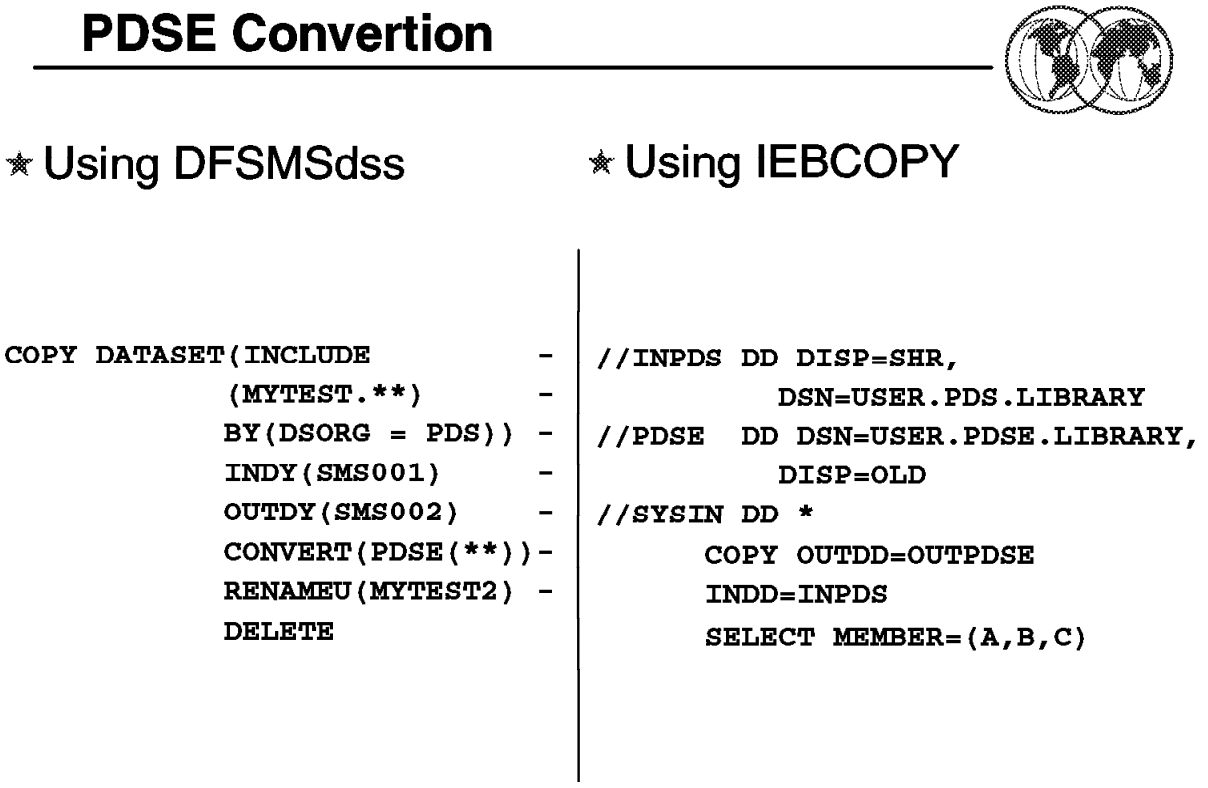


Figure 134. PDSE conversion

2.11.1 PDSE conversion

You can use IEBCOPY or DFSMSdss COPY to convert partitioned data sets to PDSEs, as shown in this visual. We recommend using DFSMSdss.

You can convert the entire data set or individual members, and also back up and restore PDSEs. By using the DFSMSdss COPY function with the CONVERT and PDS keywords, you can convert a PDSE back to a PDS. This is especially useful if you need to prepare a PDSE for migration to a site that does not support PDSEs. When copying members from a partitioned data set load-module library into a PDSE program library, or vice versa, the system invokes the program management binder.

Many types of libraries are candidates for conversion to PDSE:

- PDSs that are updated often, and that require frequent and regular reorganization
- Large PDSs that require specific device types because of the size of allocation

Converting PDSs to PDSEs is beneficial, but be aware that certain data sets are unsuitable for conversion to, or allocation as, PDSEs because the system does not retain the original block boundaries. Also, data sets requiring device dependency are inappropriate to convert or allocate because PDSEs are device-independent.

To reclaim unused space in those data sets that cannot be converted, use the DFSMSdss COMPRESS command to compress PDSs in place. This consolidates space that is no longer used within a PDS and makes it available at the end of the data set. For large or critical PDSs, you might want to copy or back up the data sets before you compress them. This maintains data set availability should the

compress fail. For more information on DFSMSdss, see *DFSMS/MVS DFSMSdss Storage Administration Reference*, SC26-4929, and *DFSMS/MVS DFSMSdss Storage Administration Guide*, SC26-4930.

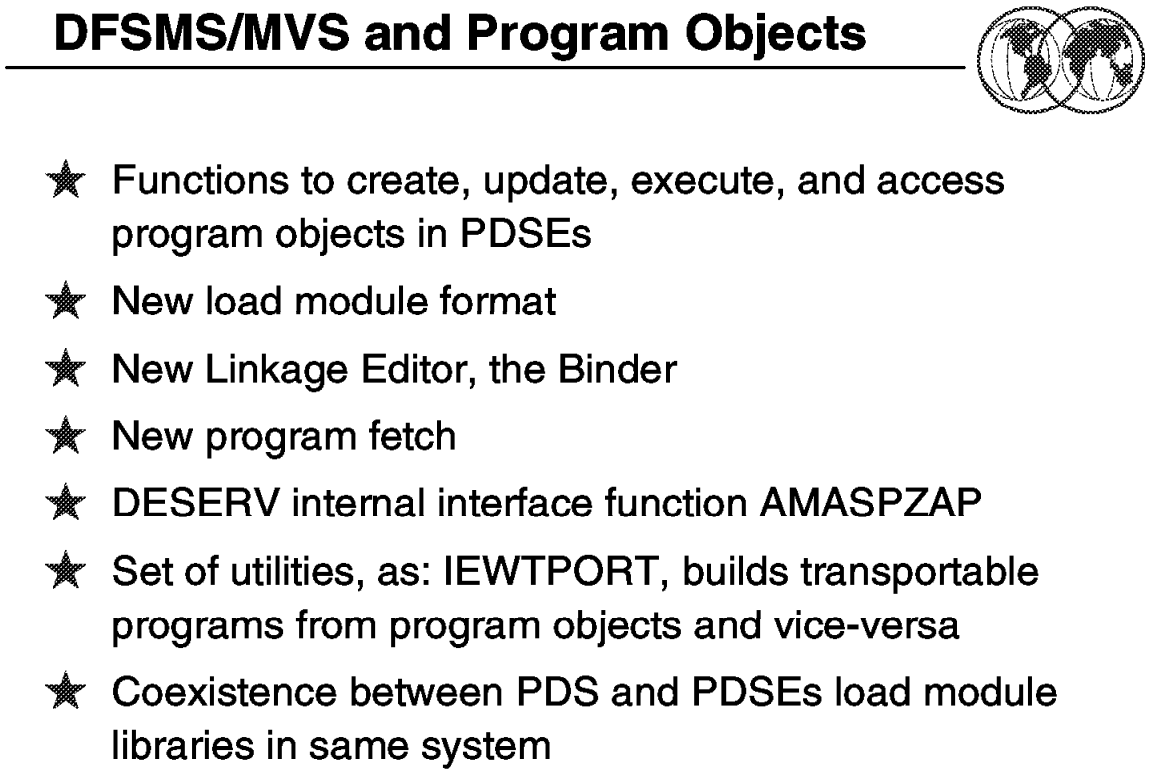


Figure 135. Program objects

2.11.2 Program objects

There are two types of PDSEs, *data* and *program object*. In this section we cover the program object type of PDSE.

The load modules stored in a PDS present some constraints, such as:

- Maximum size for TXT is 16 MB
- Maximum number of CESDs is 32 K
- All the PDS restrictions, such as:
  - It needs compression (IEBCOPY)
  - Unexpandable directory size
  - High directory search connect time (LLA and DASD cache relief)
  - Directory must be rewritten, when a member is added (sorted by collating sequence)
  - Authorization and serialization at data-set level only
  - Concurrent members creation by different tasks is an integrity exposure
- Software programs such as service aids and utilities must know the internal structures of modules and directories entries

The binder converts the output of language translators and compilers into an executable program unit that can either be read directly into virtual storage for execution or stored in a program library.

Most of the loading functions are transparent to the user. The loader will know whether the program being loaded is a load module or a program object by the source data set type. If the program is being loaded from a PDS, it calls IEWFETCH (now integrated as part of the loader) to do what it has always done. If the program is being loaded from a PDSE, a new routine is called to bring in the program using DIV. The loading is done using special loading techniques that can be influenced by externalized options.

A second directory service in support of PDSE directories, DESERV, was externalized in DFSMS/MVS 1.3. You may issue DESERV for either PDS or PDSE directory access, but you must pass the DCB address. It does not default to a pre-defined search order, as does BLDL. (Both BLDL and DESERV support ″bypass-LLA.″) DESERV returns an SMDE which, for PDSE directories, contains more information than is mapped by IHAPDS.

You create a transportable copy of the program object using IEWTPORT, then send the transportable copy to the system without program management services. A program on the target system can access the transportable copy using QSAM. If you want to load, bind, or execute a transportable program, you must first re-create the program object by executing IEWTPORT on a system with program management services installed. No programming interfaces exist to perform any of these operations on transportable programs. IEWTPORT does not support load modules, nor does it support program objects in overlay format.

To address these issues, DFSMS/MVS introduced program objects.

Program objects are a new format of load modules. In this format, load modules are called program objects. This format is only allowed when stored in a PDSE program object library.

A program object consists of *text* (executable code and data areas) and *information* about load (relocating address constants) and binding (solve externals references) in text.

The format and content of the object program and directory entry are not externalized (encapsulation).

The constraints removed from program objects are:

- Module size up to 2-Gb (TXT up to 1-Gb)
- Virtually unlimited number of aliases and external names

DFSMS/MVS has:

- Functions to create, update, execute, and access *load modules (program objects)* in PDSEs
- New load module format named *program object library*
- New linkage editor, called the binder
- New program fetch, called the loader and five new load modes. For fetching load modules, IEWFETCH is invoked by the loader
- DESERV internal interface function, to access, add, or replace directories entries in a program library (PDS or PDSE), used by:
  - Binder
  - Loader
  - LLA
  - AMASPZAP
- Set of utilities:
  - IEWTPORT, builds transportable programs from program objects and vice-versa
- Coexistence between PDS and PDSEs load module libraries in same system

Binder (DFSMS/MVS) replaces the linkage editor and loader. It executes all functions of load module linkage and editing done by linkage editor/loader. Supports new PDSE load module format program object and also supports old PDS load module format.

Program management loader is the MVS support to load program objects from PDSEs:

- Relocates all the address constants in the program to point to the appropriated areas in VS
- Supports 24-bit or 31-bit addressing
- Program objects may have different load modes based in the module characteristics and parameters specified to the binder when the object program was created (FETCHOPT). Among these load modes we have:
  - To relocate and pre-load in virtual storage before execution
  - To relocate and load in virtual storage along execution (by a kind of page fault)

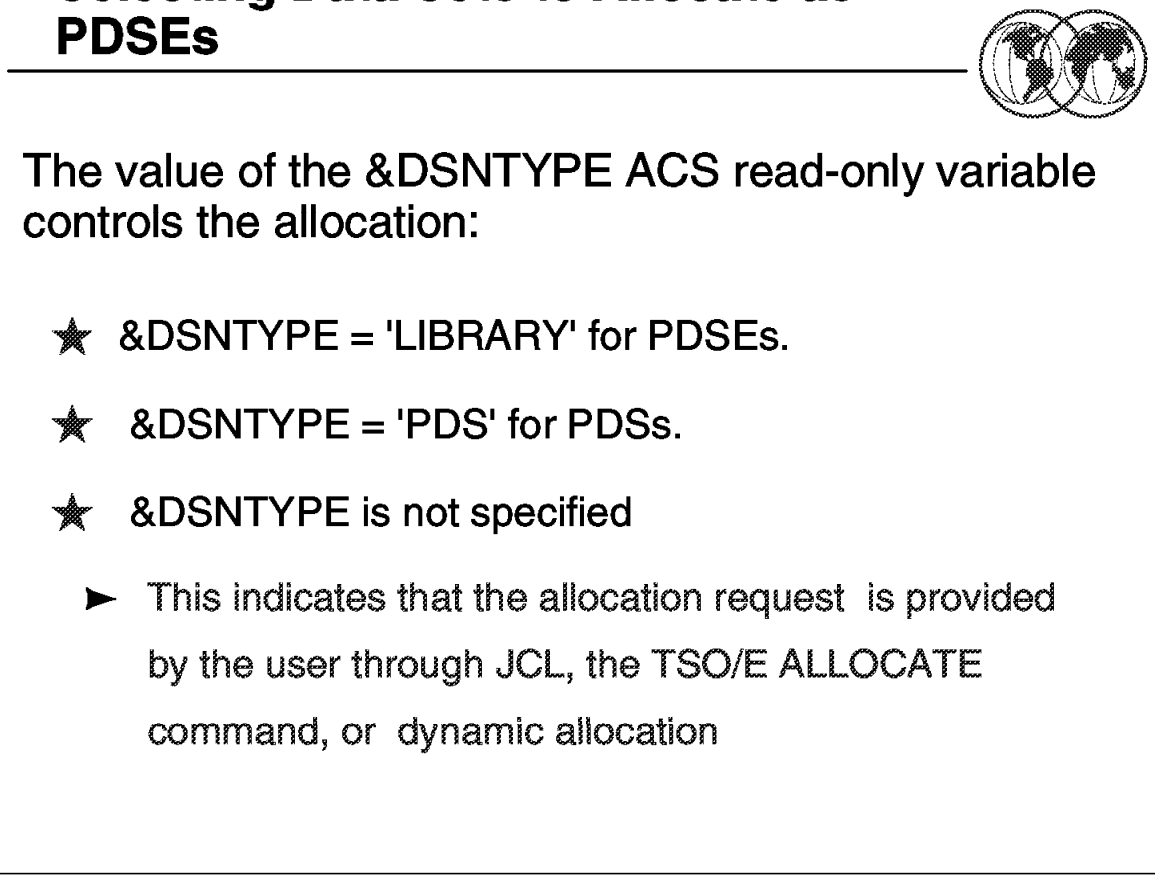


Figure 136. Selecting data sets to allocate as PDSEs

2.11.3 Selecting data sets to allocate as PDSEs

As a storage administrator, you can code appropriate ACS routines to select data sets to allocate as PDSEs and prevent inappropriate PDSs from being allocated or converted to PDSEs.

By using the &DSNTYPE read-only variable in the ACS routine for data-class selection, you can control which PDSs are to be allocated as PDSEs. The following values are valid for DSNTYPE in the data class ACS routines:

```
&DSNTYPE = 'LIBRARY' for PDSEs.
&DSNTYPE = 'PDS' for PDSs.
&DSNTYPE is not specified.  This indicates that the allocation request
is provided by the user through JCL, the TSO/E ALLOCATE command, or
dynamic allocation.
```

If you specify a DSNTYPE value in the JCL, and a different DSNTYPE value is also specified in the data class selected by ACS routines for the allocation, the value specified in the data class is ignored.

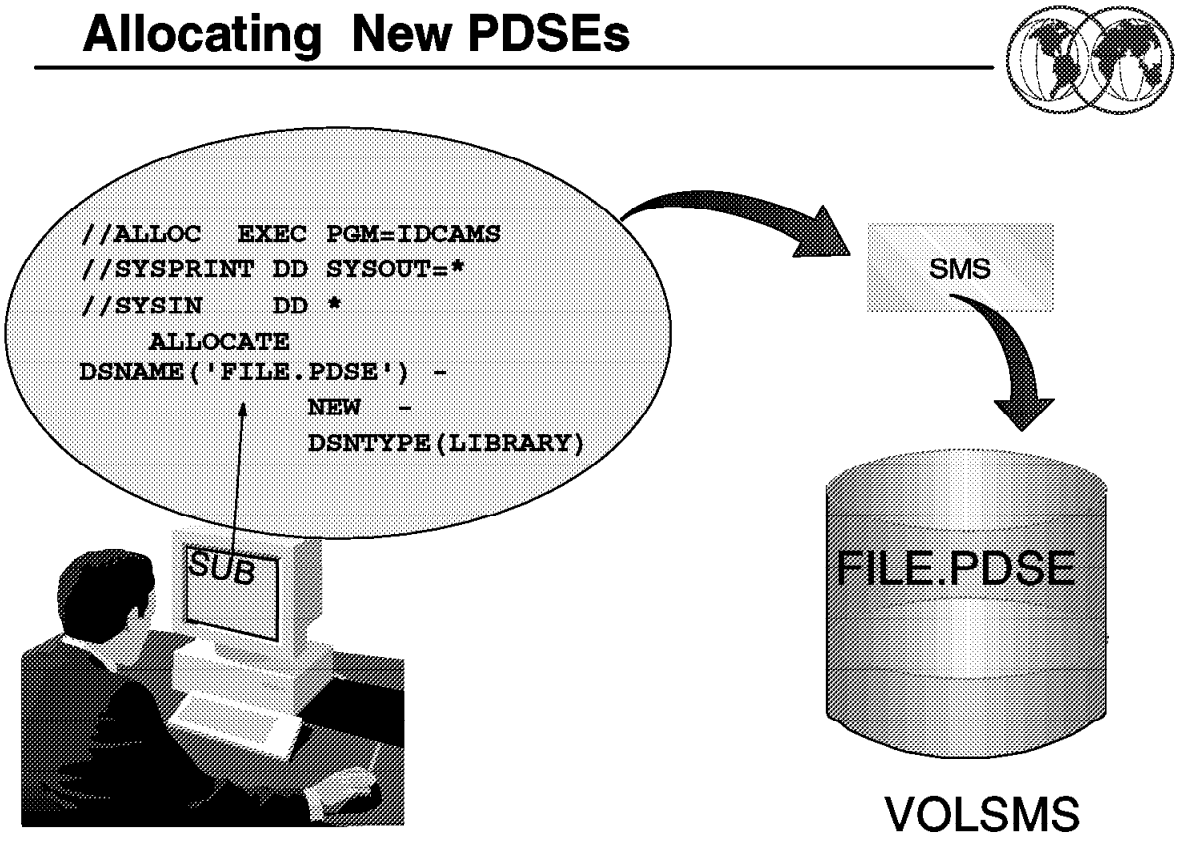


Figure 137. Allocating a new PDSE

2.11.4 Allocating new PDSEs

You can allocate PDSEs only on a system-managed volume.

Users can allocate a PDSE using the JCL keyword DSNTYPE, a data class, the ALLOCATE command, or by using IDCAMS ALLOCATE with the DSNTYPE keyword. The keyword specified is either DSNTYPE(LIBRARY) to allocate a PDSE, or DSNTYPE(PDS) to allocate a PDS. This visual shows IDCAMS ALLOCATE used with the DSNTYPE(LIBRARY) keyword to allocate a PDSE.

There are certain incompatibilities among allocation keywords. For example, you cannot use the JCL keywords DSNTYPE and RECORG (only for VSAM data sets) together.

A PDS and a PDSE can be concatenated in JCL DD statements, or by using dynamic allocation, such as the TSO ALLOCATE command.

A PDSE is allocated only if all the following criteria are met:

- Directory space or DSORG=PO is specified on the JCL statement in a dynamic allocation request, in the TSO/E ALLOCATE command, or in the data class assigned to the data set.
- The keyword DSNTYPE(LIBRARY) is specified in the command, or provided either in the data class or in the SMS installation default.
- The data set to be allocated is assigned to system-managed storage through the ACS routines.

Identifying PDSEs

```
                                DATA SET LIST
 COMMAND ===>                                          SCROLL ===> HALF
                                                    Entries 1-14 of 33
 ENTER LINE OPERATORS BELOW:                  Data Columns 27-29 of 34

  LINE                                    DATA SET     DATA SET   ENTRY
  OPERATOR         DATA SET NAME          ENVIRONMENT  NAME TYPE  TYPE
  --(1)---  ------------(2)-----------  ----(27)---  --(28)---
--(29)-
            X15PROJ.REL01.PGM           MANAGED      LIBRARY
NONVSAM
            X15PROJ.REL01.VSAM          UNMANAGED    -------    DATA
            X15PROJ.REL01.VSAM          UNMANAGED    -------    INDEX
            X15PROJ.REL01.GDS           MANAGED      -------
DEFERRE
            X15PROJ.REL01.DGTSLIB       UNMANAGED    OTHERS
NONVSAM
            X15PROJ.REL01.DGTTABL       UNMANAGED    OTHERS
NONVSAM
            X15PROJ.REL01.DUMP          UNMANAGED    -------
NONVSAM
            X15PROJ.REL01.ISPFILE       UNMANAGED    -------
```

Figure 138. Identifying a PDSE

2.11.5 Identifying PDSEs

You can use ISMF to display information associated with data set name type (DSNTYPE). This visual shows a sample data set list obtained through a catalog.

This visual lists the data set name type, where LIBRARY indicates the data set is a PDSE, and OTHERS indicates that the data set is a PDS.

The valid values for data set name type in the ISMF data set application are EXTENDED (extended sequential data sets), HFS (hierarchical file system data sets), LIBRARY (PDSEs), and OTHERS (data sets that are not allocated in the extended, HFS, or PDSE format). If you obtain the list through a catalog and column 28 contains nulls “-------” the data set name type is neither LIBRARY (PDSE) nor a PDS.

A saved data set list from a release prior to DFSMS/MVS can still be used and the values of data set name type matches those under DFSMS/MVS. For example, PDSs are indicated as OTHERS in column 28.

Introduction to Interactive Storage Management Facility (ISMF)

```
                    ISMF PRIMARY OPTION MENU - DFSMS/MVS 1.5
Enter Selection or Command ===>

Select one of the following options and press Enter:

0  ISMF Profile               - Specify ISMF User Profile
1  Data Set                   - Perform Functions Against Data Sets
2  Volume                     - Perform Functions Against Volumes
3  Management Class           - Specify Data Set Backup and Migration Criteria
4  Data Class                 - Specify Data Set Allocation Parameters
5  Storage Class              - Specify Data Set Performance and Availability
6  Storage Group              - Specify Volume Names and Free Space Thresholds
7  Automatic Class Selection  - Specify ACS Routines and Test Criteria
8  Control Data Set           - Specify System Names and Default Criteria
9  Aggregate Group            - Specify Data Set Recovery Parameters
10 Library Management         - Specify Library and Drive Configurations
11 Enhanced ACS Management    - Perform Enhanced Test/Configuration Management
C  Data Collection            - Process Data Collection Function
L  List                       - Perform Functions Against Saved ISMF Lists
R  Removable Media Manager    - Perform Functions Against Removable Media
X  Exit                       - Terminate ISMF
Use HELP Command for Help; Use END Command or X to Exit.
```

Figure 139. Introduction to ISMF

2.12 Introduction to ISMF

The Interactive Storage Management Facility (ISMF) helps you analyze and manage data and storage interactively. ISMF is an Interactive System Productivity Facility (ISPF) application. This visual shows the first ISMF panel, Primary Option Menu.

ISMF provides interactive access to the space management, backup, and recovery services of the DFSMShsm and DFSMSdss functional components of DFSMS/MVS, to the tape management services of the DFSMSrmm functional component, as well as to other products. DFSMS/MVS introduces the ability to use ISMF to define attributes of tape storage groups and libraries.

A storage administrator uses ISMF to define the installation′s policy for managing storage by defining and managing SMS classes, groups, and ACS routines. ISMF then places the configuration in an SCDS. You can activate an SCDS through ISMF or an operator command.

ISMF operates as an Interactive System Productivity Facility (ISPF) application. It is menu-driven with fast paths for many of its functions. ISMF uses the ISPF 4.2 data-tag language (DTL) to give its functional panels on workstations the look of common user access (CUA) panels and a graphical user interface (GUI).

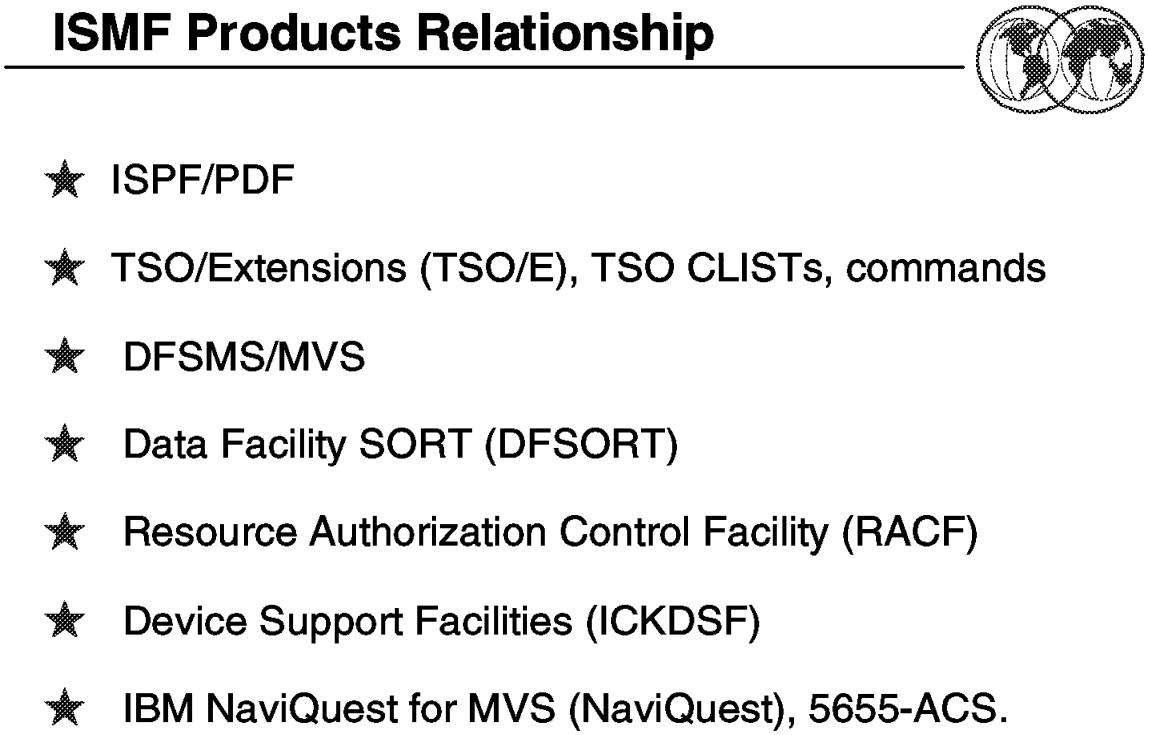


Figure 140. ISMF products relationship

2.12.1 ISMF products relationship

ISMF works with the following products that you should be familiar with:

- Interactive System Productivity Facility/Program Development Facility (ISPF/PDF), which provides the edit, browse, Data Set and Library utility functions.
- TSO/Extensions (TSO/E), TSO CLISTs and commands.
- DFSMS/MVS, which consists of four functional components, DFSMSdfp, DFSMShsf, DFSMSdss, and DFSMSrmm. ISMF is designed to use the space management and availability management (backup/recovery) functions provided by those products.
- Data Facility SORT (DFSORT), which provides the record-level functions.
- Resource Authorization Control Facility (RACF), which provides the access control function for data and services.
- Device Support Facilities (ICKDSF) to provide the storage device support and analysis functions.
- IBM NaviQuest for MVS (NaviQuest), 5655-ACS.

ISMF also works with NaviQuest, which is a new product from IBM Storage System Division Software Products. NaviQuest is a testing and reporting tool that speeds and simplifies the tasks associated with DFSMS initial implementation and ongoing ACS routine and configuration maintenance. NaviQuest assists storage administrators by allowing more automation of storage management tasks. More information on NaviQuest can be found in the NaviQuest User's Guide.

NaviQuest provides:

- Familiar ISPF panel interface to functions
- Fast, easy, bulk test-case creation
- ACS routine and DFSMS configuration-testing automation
- Storage reporting assistance
- Additional tools to aid with storage administration tasks
- Batch creation of data set and volume listings
- Printing of ISMF LISTs
- Batch ACS routine translation
- Batch ACS routine validation

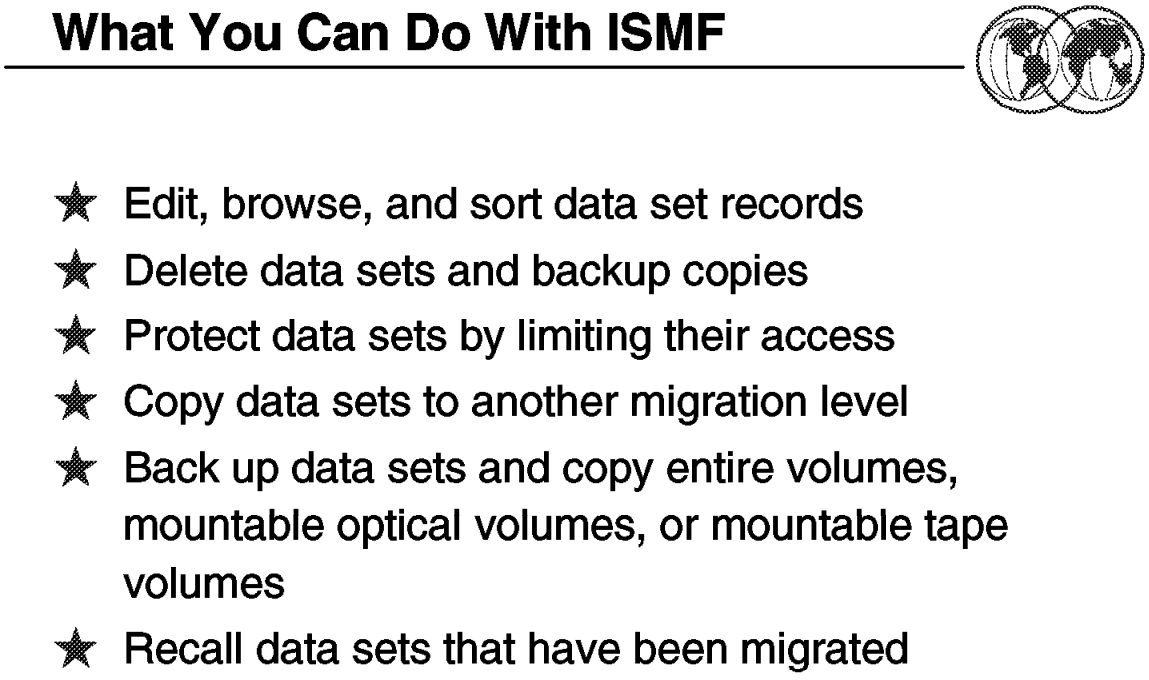


Figure 141. What you can do with ISMF

2.12.2 What you can do with ISMF

ISMF is a panel-driven interface. Use the panels in an ISMF application to:

- Display lists of information about specific data sets, DASD volumes, mountable optical volumes, and mountable tape volumes
- Generate lists of data, storage, and management classes to find out how data sets are being managed
- Display and manage lists saved from various ISMF applications

To determine which data sets appear in a data set list or which volumes appear in a volume list, you complete selection entry panels. ISMF generates a list based on your selection criteria. Once the list is built, you can use ISMF entry panels to perform space management or backup and recovery tasks against the entries in the list.

As a user performing data management tasks against individual data sets or against lists of data sets or volumes, you can use ISMF to:

- Edit, browse, and sort data set records
- Delete data sets and backup copies
- Protect data sets by limiting their access
- Recover unused space from data sets and consolidate free space on DASD volumes
- Copy data sets or DASD volumes to the same device or another device

- Migrate data sets to another migration level
- Recall data sets that have been migrated so that they can be used
- Back up data sets and copy entire volumes for availability purposes
- Recover data sets and restore DASD volumes, mountable optical volumes, or mountable tape volumes

Each site can control who can use certain functions described in this book. Your organization might require you to have authorization to use certain functions. Your security administrator can explain any restrictions your site has established.

Note: You cannot allocate data sets from ISMF. Data sets are allocated from ISPF, from TSO, or with job control language (JCL) commands. ISMF provides the DSUTIL command, which enables users to get to ISPF and toggle back to ISMF.

Accessing ISMF

```
                    ISMF PRIMARY OPTION MENU - DFSMS/MVS 1.5
Enter Selection or Command ===> _____________________________________________

Select one of the following options and press Enter:

0  ISMF Profile               - Specify ISMF User Profile
1  Data Set                   - Perform Functions Against Data Sets
2  Volume                     - Perform Functions Against Volumes
3  Management Class           - Specify Data Set Backup and Migration Criteria
4  Data Class                 - Specify Data Set Allocation Parameters
5  Storage Class              - Specify Data Set Performance and Availability
6  Storage Group              - Specify Volume Names and Free Space Thresholds
7  Automatic Class Selection  - Specify ACS Routines and Test Criteria
8  Control Data Set           - Specify System Names and Default Criteria
9  Aggregate Group            - Specify Data Set Recovery Parameters
10 Library Management         - Specify Library and Drive Configurations
11 Enhanced ACS Management    - Perform Enhanced Test/Configuration Management
C  Data Collection            - Process Data Collection Function
L  List                       - Perform Functions Against Saved ISMF Lists
R  Removable Media Manager    - Perform Functions Against Removable Media
X  Exit                       - Terminate ISMF
Use HELP Command for Help; Use END Command or X to Exit.
```

Figure 142. Accessing ISMF

2.12.3 Accessing ISMF

How you access ISMF depends on your site. You can create an option on the ISPF Primary Option Menu to access ISMF by entering the Option you specify when creating the entry. Then access ISMF by typing the characters after the arrow on the Option command line.

This starts an ISMF session from the ISPF/PDF Primary Option Menu. This visual shows the default ISMF Primary Option Menu for the storage administrator,

To access ISMF directly from TSO, use the command

```
ISPSTART PGM(DGTFMD01) NEWAPPL(DGT).
```

The ISMF Primary Option Menu shown allows you to begin the ISMF session. Figure 144 on page 225 shows the ISMF Primary Option Menu for end users.

There are two Primary Option Menus: one for end users and one for storage administrators. The menu for storage administrators includes additional applications not available to end users. Option 0, ISMF PROFILE, controls the user mode or the type of Primary Option Menu that is displayed. Refer to “Specifying a User Mode” in topic 7.2 for information on how to change the user mode.

Note: The ISMF Primary Option Menu example assumes installation of DFSMS/MVS at the current release level. For information about adding the DFSORT option to your Primary Option Menu, refer to *DFSORT Installation and Customization*, SC26-7041.

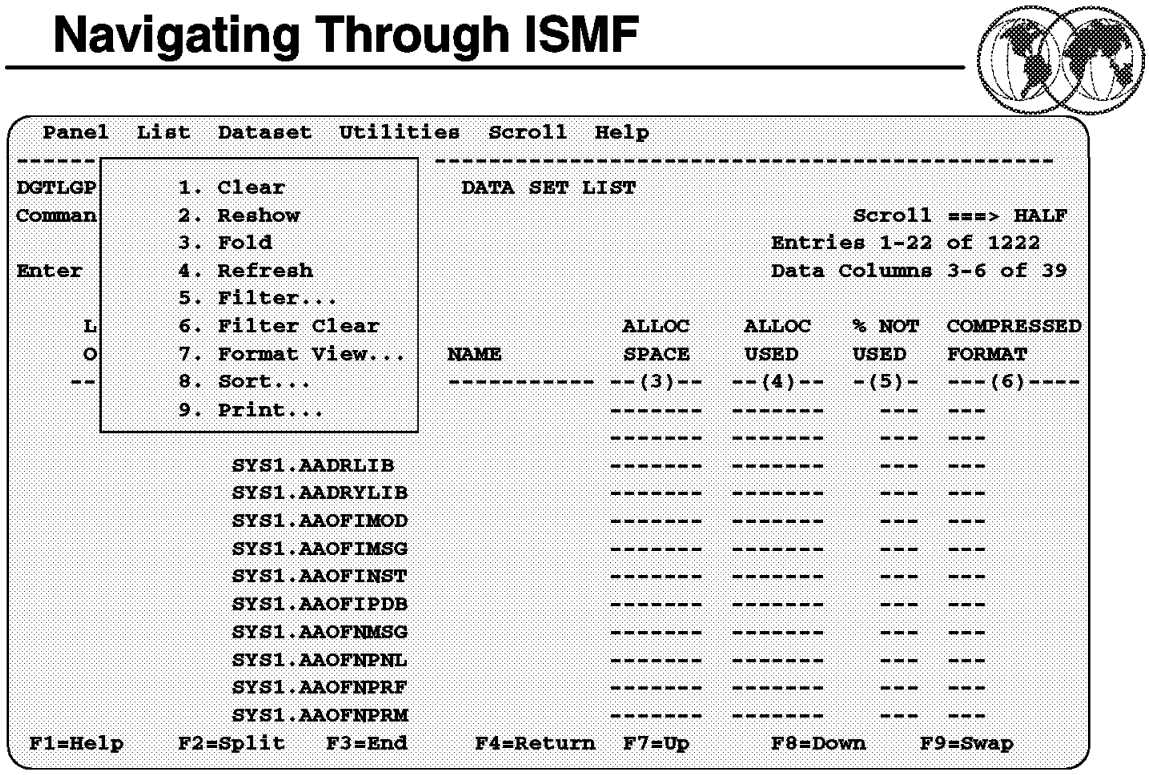


Figure 143. Navigating through ISMF

2.12.4 Navigating through ISMF

ISMF provides an action bar-driven interface that exploits many of the usability features of Common User Access (CUA) interfaces. The panels will look different than in previous releases: all screens will be mixed case and most will have action bars at the top.

2.12.4.1 Navigating through ISMF without using the action bar

You can still navigate through ISMF using the standard method of typing in a selection number and pressing Enter.

2.12.4.2 Navigating through ISMF using the action bar

Most ISMF panels have action bars at the top. The choices display in white (by default).

2.12.4.3 Using the action bar

The action bar gives you another way to move through ISMF. If the cursor is located somewhere on the panel, there are several ways to move the cursor to the action bar:

- Using the keyboard's tab key
- Using the mouse button
- Using the cursor manually

After you have chosen an action, press Enter to open the menu.

The last visual shows the List pull-down menu for the Data Set List panel. Notice the input field in the upper left corner. In the input field, type the number of the action you want, then press Enter.

ISMF Primary Option Menu

```
                          ISMF PRIMARY OPTION MENU
Enter Selection or Command ===>

Select one of the following options and press Enter:

0  ISMF Profile                    - Change ISMF User Profile
1  Data Set                        - Perform Functions Against Data Sets
2  Volume                          - Perform Functions Against Volumes
3  Management Class                - Specify Data Set Backup and Migration Criteria
4  Data Class                      - Specify Data Set Allocation Parameters
5  Storage Class                   - Specify Data Set Performance and Availability
L  List                            - Perform Functions Against Saved ISMF Lists
R  Removable Media Manager         - Perform Functions Against Removable Media
X  Exit                            - Terminate ISMF



Use HELP Command for Help; Use END Command to Exit.
```

Figure 144. ISMF Primary Option Menu

2.12.5 Selecting an option from the ISMF Primary Option menu

This visual shows the ISMF end user menu. The option 0 (ISMF Profile) controls the user mode or the type of Primary Option Menu that is displayed.

The next visual shows each option available for end users, if you need more information about the ISMF panel, refer to *DFSMS/MVS Using the Interactive Storage Management Facility*, SC26-4911.

ISMF Profile Option Menu

```
 Panel  Help
------------------------------------------------------------
DGTSPPF1                    ISMF PROFILE OPTION MENU
Enter Selection or Command ===>

Select one of the following options and Press Enter:

  0  User Mode Selection
  1  Logging and Abend Control
  2  ISMF Job Statement
  3  DFSMSdss Execute Statement
  4  ICKDSF Execute Statement
  5  Data Set Print Execute Statement
  6  IDCAMS Execute Statement
  X  Exit




Use HELP command for Help; Use END command or X to Exit.
```

Figure 145. ISMF Profile Option Menu

2.12.6 ISMF Profile Option Menu

This visual shows **Option 0 (ISMF Profile)** of the ISMF Primary Menu.

This option displays the ISMF Profile Option Menu. Use this menu to control the way ISMF runs during the session. You can:

- Change the user mode from end user to storage administrator or from storage administrator to end user.
- Control ISMF's error logging and recovery from abends.
- Define statements for ISMF to use in processing your jobs, such as:
  - JOB statements,
  - DFSMSdss
  - Device Support Facilities (ICKDSF)
  - Access Method Services (IDCAMS)
  - PRINT execute statements in your profile.

  You can select ISMF or Interactive System Productivity Facility (ISPF) JCL statements for processing batch jobs.

Data Set Selection Entry Panel

```
  Panel  Defaults  Utilities  Scroll  Help
-------------------------------------------------------------------------------
DGTDDS1                DATA SET SELECTION ENTRY PANEL              Page 1 of 5
Command ===>

For a Data Set List, Select Source of Generated List  . . 2  (1 or 2)

  1  Generate from a Saved List          Query Name To
       List Name  . .                    Save or Retrieve

  2  Generate a new list from criteria below
       Data Set Name . . . SYS1.**
        Specify Source of the new list  . . 2     (1 - VTOC, 2 - Catalog)
       1 Generate list from VTOC
          Volume Serial Number . . .             (fully or partially specified)
       2 Generate list from Catalog
          Catalog Name . . .
            Catalog Password . . . .             (if password protected)
          Volume Serial Number . . .             (fully or partially specified)
          Acquire Data from Volume . . . . . . . N  (Y or N)
          Acquire Data if DFSMShsm Migrated  . . N  (Y or N)
Use ENTER to Perform Selection; Use DOWN Command to View next Selection Panel;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 146. Data Set Selection Entry Panel

2.12.7 Data Set Selection Entry Panel

This visual shows **Option 1 (Data Set)** in the ISMF Primary Menu.

The Data Set Application constructs a list of data sets, using the filters you provide. The next visual shows the Data Set List generated by this panel.

Data Set List ISMF Panel

```
Command ===>                                                Scroll ===> HALF
                                                          Entries 1-18 of 32
Enter Line Operators below:                               Data Columns 3-6 of 39

     LINE                                      ALLOC    ALLOC   % NOT  COMPRESSED
     OPERATOR          DATA SET NAME           SPACE    USED    USED   FORMAT
    ---(1)----  ------------(2)------------  --(3)--  --(4)--  -(5)-  ---(6)----
                TALVARO                      -------  -------    ---  ---
                TALVARO.$DASDCOM             -------  -------    ---  ---
                TALVARO.$WLM                 -------  -------    ---  ---
                TALVARO.BRODCAST             -------  -------    ---  ---
                TALVARO.EOXMBMGR.EOXBMRK     -------  -------    ---  ---
                TALVARO.HCD.MSGLOG           -------  -------    ---  ---
                TALVARO.HCD.TERM             -------  -------    ---  ---
                TALVARO.HCD.TRACE            -------  -------    ---  ---
                TALVARO.LECDAILY.SCRIPT      -------  -------    ---  ---
                TALVARO.LECPGSWP.SCRIPT      -------  -------    ---  ---
                TALVARO.LECSRM.SCRIPT        -------  -------    ---  ---


 F1=Help    F2=Split   F3=End     F4=Return  F7=Up      F8=Down    F9=Swap
F10=Left   F11=Right  F12=Cursor
```

Figure 147. Data Set List Panel

2.12.8 Data Set List panel

Use line operators to do tasks with individual data sets. Use list commands to do tasks with a group of data sets. These tasks include editing, browsing, recovering unused space, copying, migrating, deleting, backing up, and restoring of data sets. TSO commands and CLISTs can also be used as line operators or list commands. You can save a copy of a data set list and reuse it later.

If ISMF is unable to get certain information required to check if a data set meets the selection criteria specified, that data set is also be included in the list. This is indicated by dashes on the corresponding column. For example, if ISMF is unable to check if a data set meets the specified volume serial number criteria, that data set still appears in the list with dashes in the corresponding Volume Serial Number field.

Volume List Selection Menu

```
 Panel  Help
------------------------------------------------------------------------------
DGTSMOV1                  VOLUME LIST SELECTION MENU
Enter Selection or Command ===>

1  DASD                        - Generate a List of DASD Volumes
2  Mountable Optical           - Generate a List of Mountable Optical Volumes
3  Mountable Tape              - Generate a List of Mountable Tape Volumes









Use HELP Command for Help; Use END Command to Exit.
```

Figure 148. Volume List Selection Menu

2.12.9 Volume List Selection Menu

The **Option 2 (Volume)** in the ISMF Primary Menu take us to the menu shown in this visual. If we pick up DASD, we will see the Volume Selection Entry Panel, where using filters you can select a Volume List Panel, shown in the next visual.

ISMF Volume List Panel

```
Command ===>                                                Scroll ===> PA
                                                          Entries 1-22 of 67
Enter Line Operators below:                               Data Columns 3-8 of


    LINE       VOLUME  FREE     %     ALLOC    FRAG   LARGEST  FREE
    OPERATOR   SERIAL  SPACE    FREE  SPACE    INDEX  EXTENT   EXTENTS
    ---(1)---- -(2)--  --(3)--  (4)-  --(5)--  -(6)-  --(7)--  --(8)--
               DFPLIB  -------   ---  -------    ---  -------      101
               DFPTL3  -------   ---  -------    ---  -------      280
               ISMF01  -------   ---  -------    ---  -------      620
               ISMF02  -------   ---  -------    ---  -------      820
               LIBTST  -------   ---  -------    ---  -------      336
               PAGEIC  -------   ---  -------    ---  -------      118
               PAGE02  -------   ---  -------    ---  -------      182
               SMSDIV  -------   ---  -------    ---  -------      712
               SPOOL1  -------   ---  -------    ---  -------    -----
               SYSLIB  -------   ---  -------    ---  -------    -----
               SYSRES  -------   ---  -------    ---  -------    -----
```

Figure 149. ISMF Volume List panel.

2.12.10 ISMF Volume List panel

The volume application constructs a list of DASD volumes, mountable optical volumes, or mountable tape volumes. Use line operators to do tasks with an individual volume. These tasks include consolidating or recovering unused space, copying, backing up, and restoring volumes. TSO commands and CLISTs can also be line operators or list commands. You can save a copy of a volume list and reuse it later. With the list of mountable optical volumes or mountable tape volumes, you can only browse the list. This option displays the Volume List Selection Menu. For information about when to select the Volume option and tasks you can do using the Volume Application, see Chapter 3, ″Generating Lists″ in topic 3.0, Chapter 4, ″Using the Data Set or Volume List″ in topic 4.0, and Chapter 6, ″Performing Data and Storage Management Tasks″ in topic 6.0. in *DFSMS/MVS Using the Interactive Storage Management Facility*, SC26-4911.

Management Class Application Selection

```
 Panel  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTSCMC2            MANAGEMENT CLASS APPLICATION SELECTION        Page 1 of 2
Command ===>

To perform Management Class Operations, Specify:
  CDS Name  . . . . . . . . . . 'ACTIVE'
                                (1 to 44 character data set name or 'Active' )
  Management Class Name . . . *          (For Management Class List, fully or
                                         partially specified or * for all)

Select one of the following options  :
  1  1. List     - Generate a list of Management Classes
     2. Display - Display a Management Class
     3. Define  - Define a Management Class
     4. Alter   - Alter a Management Class

If List Option is chosen,
   Enter "/" to select option      Respecify View Criteria
                                   Respecify Sort Criteria

 Use ENTER to Perform Selection; Use DOWN Command to View next Selection Panel;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 150. Management Class Application Selection

2.12.11 Management Class Application Selection

This visual shows **Option 3 (Management Class)** in the ISMF Primary Menu

This panel allows you to display, modify, and define options for the management classes of the storage management subsystem (SMS). It also constructs a list of the available management classes.

Next visual shows the management class list generated by the filters located in this panel with the option Display.

ISMF Management Class List

```
 Command ===>                                                Scroll ===> HALF
                                                          Entries 1-19 of 20
                                                          Data Columns 3-7 of 40
CDS Name : ACTIVE

Enter Line Operators below:

     LINE       MGMTCLAS EXPIRE     EXPIRE        RET    PARTIAL       PRIMARY
     OPERATOR   NAME     NON-USAGE  DATE/DAYS     LIMIT  RELEASE       DAYS
     ---(1)---- --(2)--- ---(3)---  ---(4)----   --(5)-- ----(6)----   ---(7)--
                BYRON      NOLIMIT     NOLIMIT  NOLIMIT  NO                 2
                CICSLSMC   NOLIMIT     NOLIMIT  NOLIMIT  NO              ----
                CICSRLSM   NOLIMIT     NOLIMIT  NOLIMIT  NO              ----
                DBML2      NOLIMIT     NOLIMIT  NOLIMIT  COND_IMMED         2
                DBSTAN     NOLIMIT     NOLIMIT  NOLIMIT  NO                15
                EXTBAK     NOLIMIT     NOLIMIT  NOLIMIT  COND_IMMED        15
                GDGBKUP    NOLIMIT     NOLIMIT  NOLIMIT  YES_IMMED          2
                GDGPROD    NOLIMIT     NOLIMIT  NOLIMIT  YES_IMMED         15
                INTERIM          3           3        3  YES_IMMED          3
                LS@CICS    NOLIMIT     NOLIMIT  NOLIMIT  NO              ----
                MCABARS    NOLIMIT     NOLIMIT  NOLIMIT  NO                 2
                MCGEOFF        100     NOLIMIT  NOLIMIT  YES                10
```

Figure 151. ISMF management class list

2.12.12 ISMF management class list

This visual shows the partial contents of the ISMF management class list panel.

Data Class Application Selection

```
  Panel  Utilities  Help
------------------------------------------------------------------------------
DGTSCDC2               DATA CLASS APPLICATION SELECTION
Command ===>

To perform Data Class Operations, Specify:
  CDS Name . . . . . . 'ACTIVE'
                              (1 to 44 character data set name or 'Active' )
  Data Class Name  . . *        (For Data Class List, fully or partially
                                specified or * for all)

Select one of the following options  :
  1  1. List    - Generate a list of Data Classes
     2. Display - Display a Data Class
     3. Define  - Define a Data Class
     4. Alter   - Alter a Data Class

If List Option is chosen,
   Enter "/" to select option      Respecify View Criteria
                                   Respecify Sort Criteria

Use ENTER to Perform Selection;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 152. Data Class Application Selection

2.12.13 Data Class Application Selection

This visual shows **Option 4 (Data Class)** of the ISMF primary menu. In this panel you can define the way data sets are allocated.

Data class attributes are assigned to a data set when it is created. Data class attributes apply to both DFSMS-managed and non-DFSMS-managed data sets. Attributes specified in JCL or equivalent allocation statements override those specified in a data class. Individual attributes in a data class can be overridden by JCL, TSO, IDCAMS, and dynamic allocation statements.

The next visual shows the Data Class List generated by the filters located in this panel with the option Display.

ISMF Data Class List Panel

```
 Command ===>                                                Scroll ===> HALF
                                                          Entries 1-19 of 46
                                                          Data Columns 3-9 of 39
CDS Name : ACTIVE

Enter Line Operators below:

     LINE       DATACLAS                                                  AVG
     OPERATOR   NAME      RECORG  RECFM  LRECL  KEYLEN  KEYOFF  AVGREC  VALUE
     ---(1)---- --(2)---  -(3)--  -(4)-  -(5)-  -(6)--  -(7)--  -(8)--  -(9)-
                CICSLSDC  LS      ----   -----    ---   -----   U        4096
                CICSSTG   LS      ----   -----    ---   -----   U        4096
                CST       --      ----   -----    ---   -----   -        -----
                DATAF     --      FB       80     ---   -----   U          80
                DATAV     --      VB      255     ---   -----   U         255
                DCADP     --      ----   -----    ---   -----   -        -----
                DCCOMP    --      ----   -----    ---   -----   -        -----
                DCEF      --      ----   -----    ---   -----   -        -----
                DCHFS     --      ----   -----    ---   -----   -        -----
                DCPDSE    --      ----   -----    ---   -----   -        -----
                DCSMB     KS      ----   -----    ---   -----   -        -----
                DCSTRIPE  --      ----   -----    ---   -----   -        -----
```

Figure 153. ISMF data class list

2.12.14 ISMF data class list

This visual shows the output of the data class list panel.

Storage Class Application Selection

```
  Panel  Utilities  Help
------------------------------------------------------------------------------
DGTSCSC1              STORAGE CLASS APPLICATION SELECTION
Command ===>

To perform Storage Class Operations, Specify:
  CDS Name  . . . . . . . 'ACTIVE'
                          (1 to 44 character data set name or 'Active' )
  Storage Class Name  . . *         (For Storage Class List, fully or
                                    partially specified or * for all)

Select one of the following options  :
  1  1. List          - Generate a list of Storage Classes
     2. Display       - Display a Storage Class
     3. Cache Display - Display Storage Classes/Cache Sets

If List Option is chosen,
   Enter "/" to select option      Respecify View Criteria
                                   Respecify Sort Criteria
If Cache Display is Chosen, Specify Cache Structure Name . .

Use ENTER to Perform Selection;
Use HELP Command for Help; Use END Command to Exit;
```

Figure 154. Storage Class Application Selection

2.12.15 Storage Class Application Selection

This visual shows **Option 5 (Storage Class)** of the ISMF primary menu.

The Storage Class Application Selection panel lets the storage administrator specify performance objectives and availability attributes that characterize a collection of data sets. For objects, the storage administrator can define the performance attribute Initial Access Response Seconds. A data set or object must be assigned to a storage class in order to be managed by DFSMS.

The next visual shows the Storage Class List generated by the filters located in this panel with the option Display.

ISMF Storage Class List Panel

```
                    STORAGE CLASS LIST
Command ===>                              Scroll ===> HALF
                          Entries 1-19 of 37
                          Data Columns 3-7 of 18
CDS Name : ACTIVE

Enter Line Operators below:

 LINE       STORCLAS   DIR RESP   DIR     SEQ RESP   SEQ
 OPERATOR   NAME       (MSEC)     BIAS      (MSEC)   BIAS   AVAILABILITY
 ---(1)---- --(2)---   --(3)---   (4)-     --(5)---  (6)-   -------(7)--------
            CICSLSSC      ---      -         ---     -       NOPREF
            CICSRLS         5      W           5     W       NOPREF
            CICSSTG       ---      -         ---     -       NOPREF
            CRITICAL       10      -          10     -       CONTINUOUS
            DBCRIT         10      W          10     R       CONTINUOUS
            FAST            5      -           5     -       NOPREF
            FASTREAD        5      R           5     R       NOPREF
            FASTWRIT        5      W           5     W       NOPREF
            GSPACE        ---      -         ---     -       NOPREF
```

Figure 155. ISMF Storage Class List

2.12.16 ISMF storage class List

The visual shows the Storage Class List panel.

You can specify the DISPLAY line operator next to any class name on a class list to generate a panel that displays values associated with that particular class. This information can help you decide whether you need to assign a new DFSMS class to your data set or object.

If you determine that a data set you own should be associated with a different management class or storage class, and if you have authorization, you can use the ALTER line operator against a data set list entry to specify another storage class or management class.

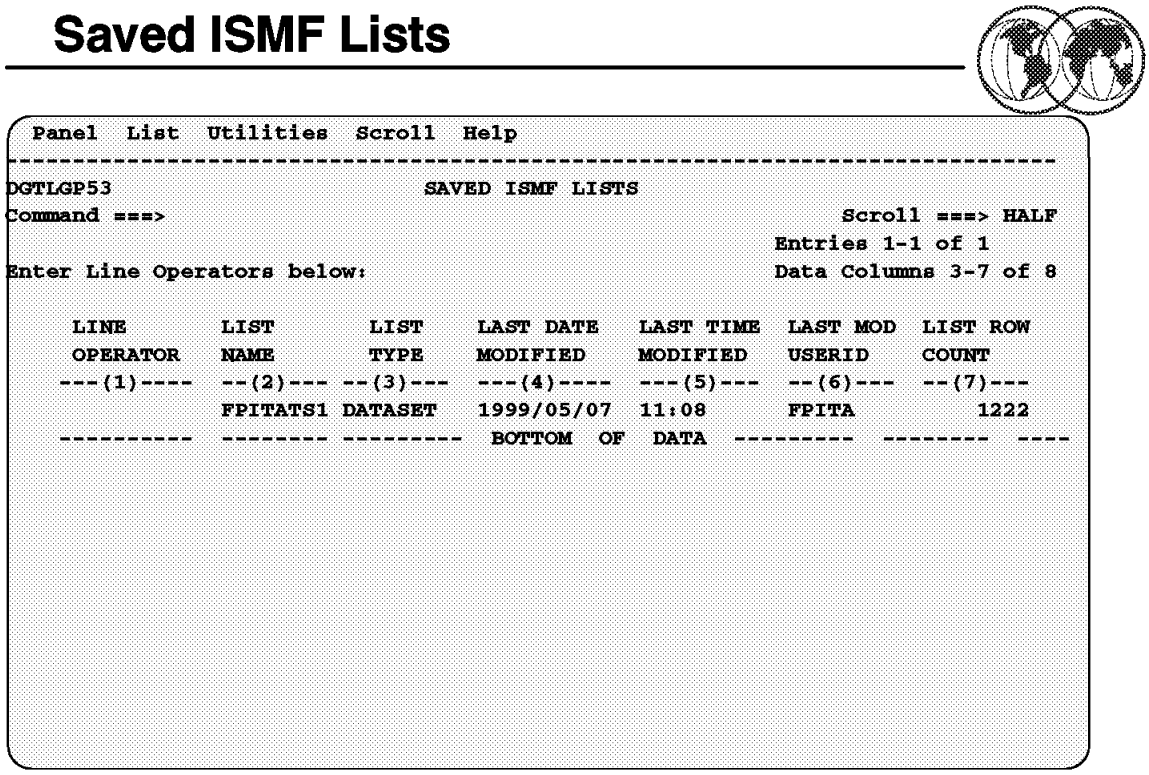


Figure 156. Saved ISMF Lists

2.12.17 Saved ISMF Lists

This visual shows **Option L (Lists)** of the ISMF primary menu.

The List Application displays a list of all lists saved from ISMF applications. Each entry in the list represents a list that was saved. When you select the List option, the Saved ISMF Lists Panel is displayed if there are any saved ISMF lists. If there are no saved lists to be found, the Primary Option Menu is re-displayed with the message that the list is empty.

You can reuse and delete saved lists. From the List Application, you can reuse lists as if they were created from the corresponding application. You can then use line operators and commands to tailor and manage the information in the saved lists.

Removable Media Manager

```
 Panel  Help
------------------------------------------------------------------------------
EDG@PRIM              REMOVABLE MEDIA MANAGER (DFSMSrmm)
Option ===>

0  OPTIONS       - Specify dialog options and defaults
1  USER          - General user facilities
2  LIBRARIAN     - Librarian functions
3  ADMINISTRATOR - Administrator functions
4  SUPPORT       - System support facilities
5  COMMANDS      - Full DFSMSrmm structured dialog
6  LOCAL         - Installation defined dialog
X  EXIT          - Exit DFSMSrmm Dialog

Enter selected option or END command.  For more info., enter HELP or PF1.




5695-DF1 (C) COPYRIGHT IBM CORPORATION 1993
```

Figure 157. Removable Media Manager (DFSMSrmm)

2.13 Removable Media Manager (DFSMSrmm)

This visual shows **Option R (Lists)** of the ISMF primary menu. This option displays the Primary Option Menu of the Removable Media Manager application.

Under normal circumstances, the DFSMSrmm subsystem starts automatically through IPL procedures, either standard or as modified by your installation. In exceptional cases, such as after recovery of the DFSMSrmm control data set, you might need to restart the subsystem.

Data Facility Removable Media Manager for MVS/DFP Version 3 (DFRMM) Program Offering provides support for non-system-managed tape libraries. When you use DFRMM and DFSMSrmm together, or multiple DFSMSrmm systems, they can share the same control data set. When both DFRMM and DFSMSrmm share the control data set, you can use the DFRMM ISPF dialog and RMM TSO subcommands to display all information that has been recorded in the control data set. There are some restrictions on using the RMM TSO subcommands from DFRMM, and from DFSMSrmm on a non-system-managed tape system, to add and change information in the control data set.

Chapter 3. System Modification Program/Enhanced (SMP/E)

OS/390 SMP/E is a tool designed to manage the installation of software products on your OS/390 system and to track the modifications you make to those products. Usually, it is the system programmer's responsibility to ensure that all software products and their modifications are properly installed on the system. The system programmer also has to ensure that all products are installed at the proper level so all elements of the system can work together. At first, that might not sound too difficult, but as the complexity of the software configuration increases, so does the task of monitoring all the elements of the system. To better understand this, let's take a closer look at the OS/390 system and see how SMP/E can help you maintain it.

Over time, you may need to change some of the elements of your system. These changes may be necessary to improve the usability or reliability of a product. You may want to add some new functions to your system, upgrade some of the elements of your system, or modify some elements for a variety of reasons. In all cases, you are making system modifications. In SMP/E, we refer to these system modifications as SYSMODs.

A SYSMOD is the actual package containing information SMP/E needs to install and track system modifications. SYSMODs are composed of two parts:

- Modification control statements (MCS), designated by ++ as the first two characters, that tell SMP/E:
  - What elements are being updated or replaced
  - How the SYSMOD relates to product software and other SYSMODs
  - Other specific installation information
- Modification text, which is the object modules, macros, and other elements supplied by the SYSMOD

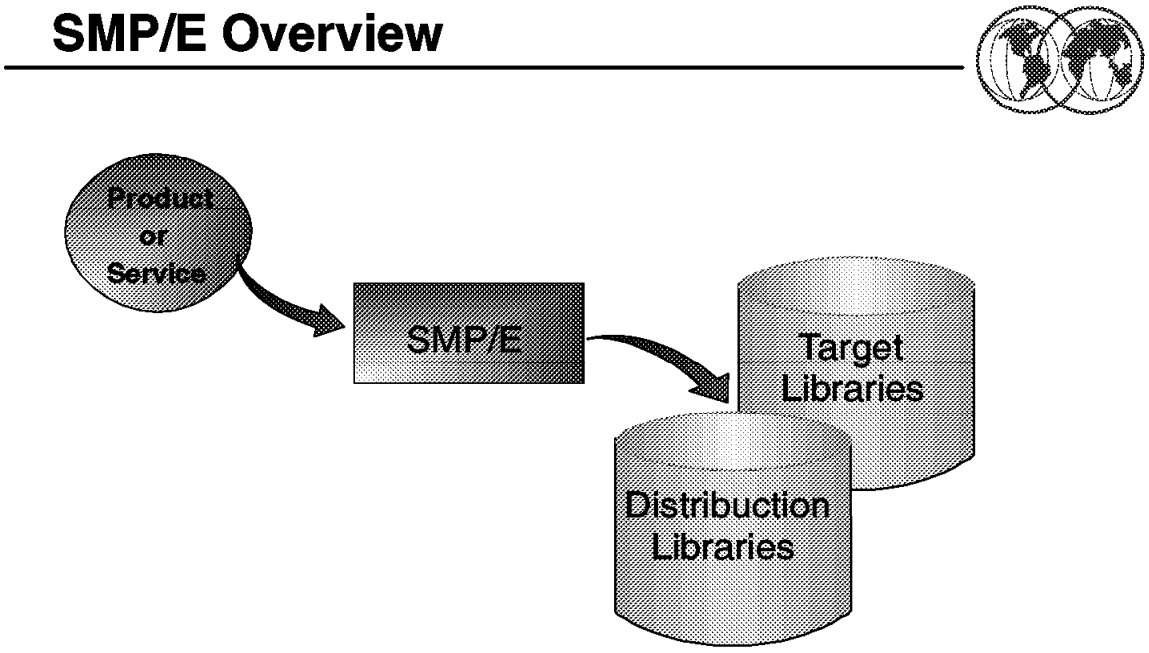


Figure 158. SMP/E overview

3.1 Introduction to SMP/E

SMP/E is the basic tool for installing and maintaining software in OS/390 systems and subsystems. It controls these changes at the element level by:

- Selecting the proper levels of elements to be installed from a large number of potential changes
- Calling system utility programs to install the changes
- Keeping records of the installed changes

SMP/E is an integral part of the installation, service, and maintenance processes for CBPDOs, ProductPacs, ServicePacs, and selective follow-on service for CustomPacs. In addition, SMP/E can be used to install and service any software that is packaged in SMP/E system modification (SYSMOD) format.

SMP/E can be run either using batch jobs or using dialogs under Interactive System Productivity Facility/Program Development Facility (ISPF/PDF). SMP/E dialogs help you interactively query the SMP/E database as well as create and submit jobs to process SMP/E commands.

These are some of the types of software that can be installed by SMP/E:

- Products and service provided in CBPDOs and CustomPac offerings
- Products and service from IBM Software Distribution Centers not provided in CBPDOs or CustomPac offerings
- Other products and service

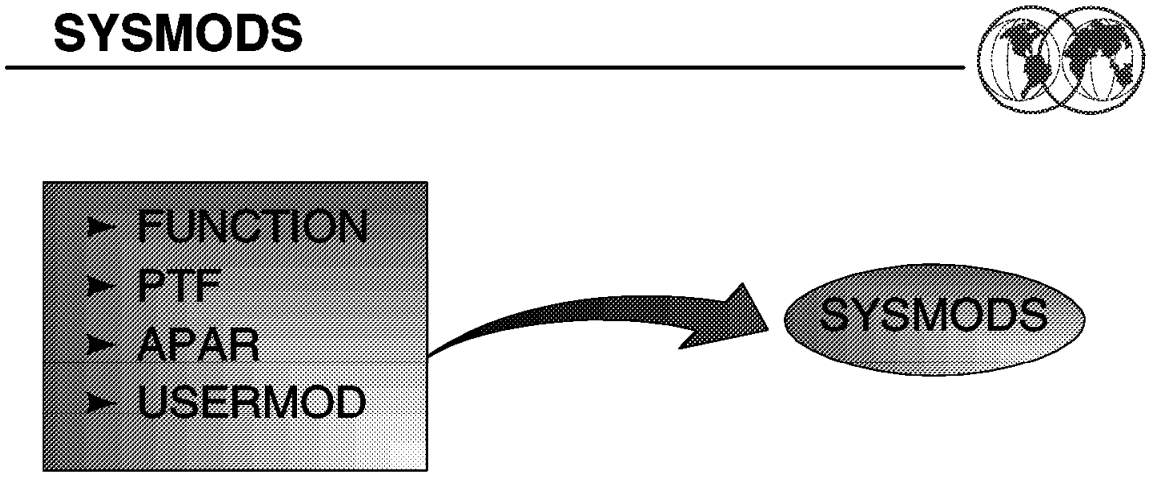


Figure 159. SYSMODs

3.2 SYSMODs

Over time, you may need to change some of the elements of your system. These changes may be necessary to improve the usability or reliability of a product. You may want to add some new functions to your system, upgrade some of the elements of your system, or modify some elements for a variety of reasons. In all cases, you are making system modifications. In SMP/E, we refer to these system modifications as SYSMODs.

SYSMOD is the actual package containing information SMP/E needs to install and track system modifications. SYSMODs are composed of two parts:

- Modification control statements (MCS), designated by ++ as the first two characters, that tell SMP/E:
  1. What elements are being updated or replaced
  2. How the SYSMOD relates to product software and other SYSMODs
  3. Other specific installation information
- Modification text, which is the object modules, macros, and other elements supplied by the SYSMOD

There are four different categories of SYSMODs, each supporting a task you might want to perform:

Function Introduce the elements for a product.

PTF PTF (program temporary fix) prevents or fixes problems with an element, or introduces new elements.

APAR APAR (authorized program analysis reports) fixes problems with an element.

USERMOD USERMOD (user modifications) customizes an element.

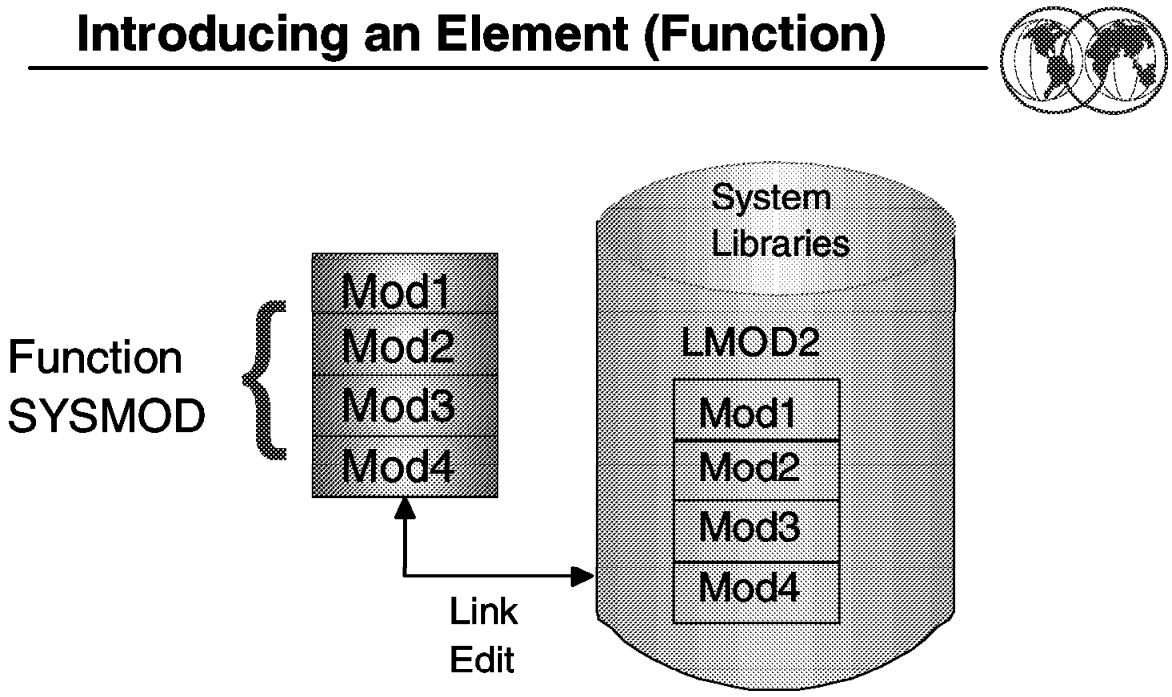


Figure 160. Introducing an element

3.2.1 Introducing an element - the function SYSMOD

One way you can modify your system is to introduce new elements into that system. To accomplish this with SMP/E, you can install a function SYSMOD. The function SYSMOD introduces a new product, a new version or release of a product, or updated functions for an existing product into the system. All other types of SYSMODs are dependent upon the function SYSMOD, because they are all modifications of the elements originally introduced by the function SYSMOD.

When we refer to installing a function SYSMOD, we are referring to the placing of all the product's elements in the system data sets, or libraries. Examples of these libraries are SYS1.LINKLIB, SYS1.LPALIB, and SYS1.SVCLIB. This visual shows the process of creating executable code in the production system libraries.

In the figure, the installation of a function SYSMOD link-edits object modules Mod1, Mod2, Mod3, and Mod4 to create load module LMOD2. The executable code created in load module LMOD2 is installed in the system libraries through the installation of the function SYSMOD.

There are two types of function SYSMODs:

- A *base* function SYSMOD adds or replaces an entire system function. Examples of base functions are SMP/E and JES3.
- A *dependent* function SYSMOD provides an addition to an existing system function. It is called dependent because its installation depends upon a base function already being installed. Examples of dependent functions are the language features for SMP/E.

Both base function SYSMODs and dependent function SYSMODs are used to introduce new elements into the system.

Figure 161 shows an example of a simple function SYSMOD that introduces four elements:

```
++FUNCTION(FUN0001)         /* SYSMOD type and identifier.   */.
++VER(Z038)                 /* For an OS/390 system          */.
++MOD(MOD1)  RELFILE(1)     /* Introduce this module         */
             DISTLIB(AOSFB) /* in this distribution library. */.
++MOD(MOD2)  RELFILE(1)     /* Introduce this module         */
             DISTLIB(AOSFB) /* in this distribution library. */.
++MOD(MOD3)  RELFILE(1)     /* Introduce this module         */
             DISTLIB(AOSFB) /* in this distribution library. */.
++MOD(MOD4)  RELFILE(1)     /* Introduce this module         */
             DISTLIB(AOSFB) /* in this distribution library. */.
```

Figure 161. Example SYSMOD with four elements

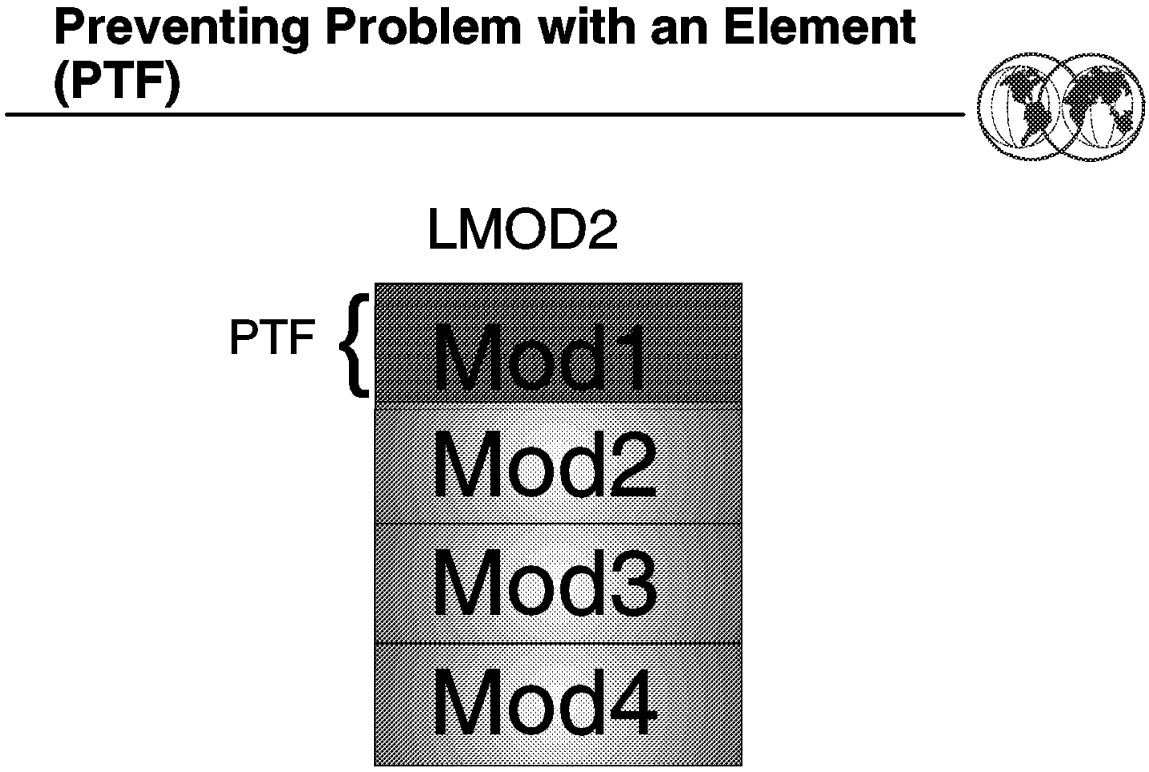


Figure 162. Preventing problems with an element (PTF)

3.2.2 Preventing problems with an element (PTF)

When a problem with a software element is discovered, IBM supplies its customers with a tested fix for that problem. This fix comes in the form of a program temporary fix (PTF). Although you may not have experienced the problem the PTF is intended to prevent, it is wise to install the PTF on your system. The PTF SYSMOD is used to install the PTF, thereby preventing the occurrence of that problem on your system.

Usually, PTFs are designed to replace or update one or more complete elements of a system function.

In the visual, we see a previously installed load module, LMOD2. If we want to replace the element Mod1, we should install a PTF SYSMOD that contains the module Mod1. That PTF SYSMOD replaces the element in error with the corrected element. As part of the installation of the PTF SYSMOD, SMP/E relinks LMOD2 to include the new and corrected version of Mod1.

Figure 163 on page 246 shows an example of a simple PTF SYSMOD:

```
++PTF(PTF0001)              /* SYSMOD type and identifier.   */.
++VER(Z038) FMID(FUN0001)   /* Apply to this product.        */.
++MOD(MOD1)                 /* Replace this module           */
            DISTLIB(AOSFB)  /* in this distribution library. */.
...
... object code for module
 ...
```

Figure 163. Example simple PTF SYSMOD

PTF SYSMODs are always dependent upon the installation of a function SYSMOD. In some cases, some PTF SYSMODs may also be dependent upon the installation of other PTF SYSMODs. These dependencies are called *prerequisites*.

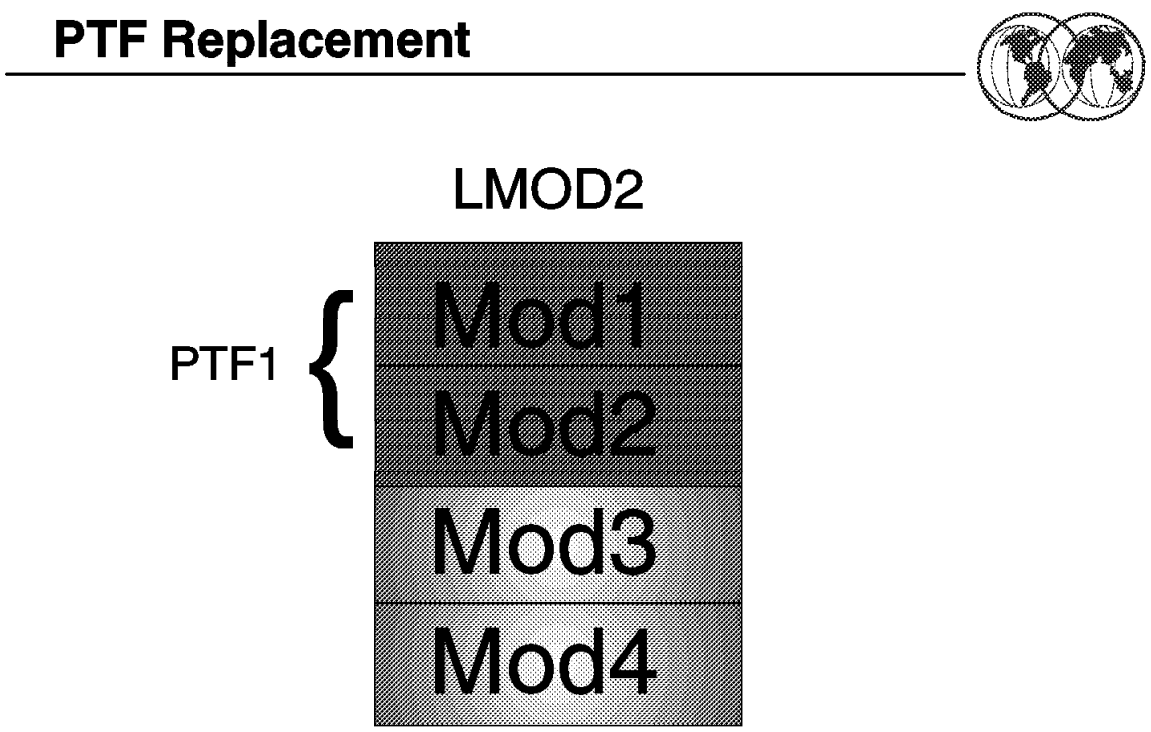


Figure 164. PTF replacement

3.2.2.1 PTF replacement

The importance of keeping track of system elements and their modification becomes readily apparent when we examine the OS/390 maintenance process. Often, a PTF contains multiple element replacements. In the visual PTF1 contains replacements for two modules, Mod1 and Mod2. Although load module LMOD2 contains four modules, only two of those modules are being replaced.

But what happens if a second PTF replaces some of the code in a module that was replaced by PTF1?

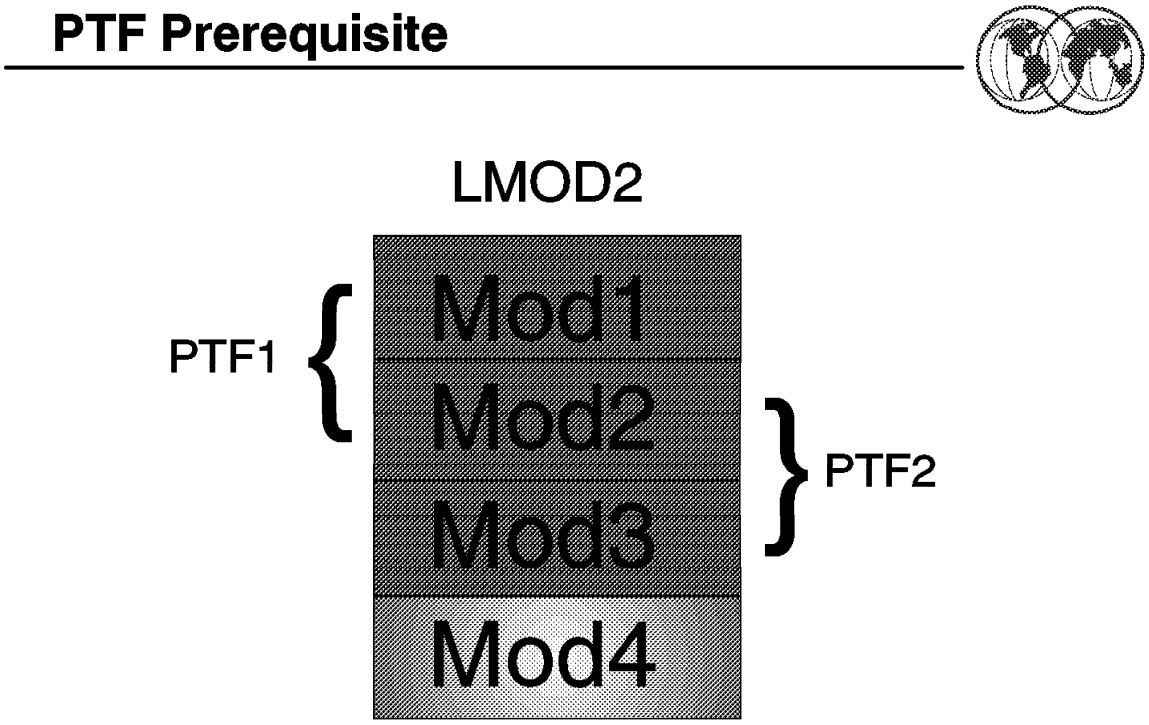


Figure 165. PTF prerequisite

3.2.2.2 PTF prerequisite

In this example, PTF2 contains replacements for Mod2 and Mod3. For Mod1, Mod2, and Mod3 to interface successfully, PTF1 must be installed before PTF2. That′s because Mod3 supplied in PTF2 may depend on the PTF1 version of Mod1 to be present. It is this dependency that constitutes a prerequisite. SYSMOD prerequisites are identified in the modification control statements (MCS) part of the SYSMOD package

In addition to tracking prerequisites, there is another important reason to track system elements. The same module is often part of many different load modules.

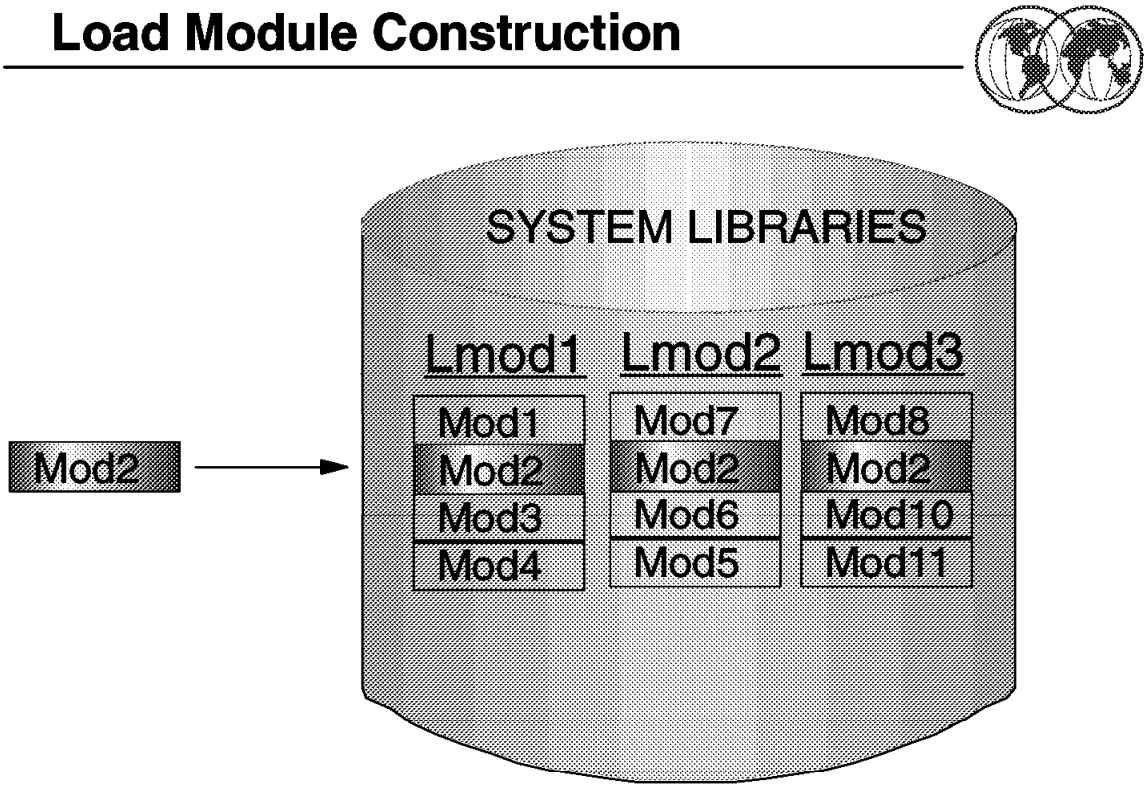


Figure 166. Load module construction

3.2.2.3 Load module construction

In the visual, the same Mod2 module is present in LMOD1, LMOD2, and LMOD3. When a PTF is introduced that replaces the element Mod2, that module must be replaced in all the load modules in which it exists. Therefore, it is imperative that we keep track of all load modules and the modules they contain.

You can now appreciate how complicated the tracking of system elements and their modification levels can become. Let′s take a brief look at how we implement the tracking capabilities of SMP/E.

3.2.2.4 Tracking and controlling requisites

To track and control elements successfully, all elements and their modifications and updates must be clearly identified to SMP/E. SMP/E relies on modification identifiers to accomplish this. There are three modification identifiers associated with each element:

- *Function modification identifiers (FMIDs)* that identify the function SYSMOD that introduced the element into the system.
- *Replacement modification identifiers (RMIDs)* that identify the last SYSMOD (usually a PTF SYSMOD) to replace the element.
- *Update modification identifiers (UMIDs)* that identify the SYSMODs that have updated an element since it was last replaced.

SMP/E uses these modification identifiers to track all SYSMODs installed on your system. This ensures that they are installed in the proper sequence. Now that we realize the need for element tracking and know the types of things SMP/E tracks, we will now look at how APARs are used.

Fixing a Problem with an Element The APAR SYSMOD

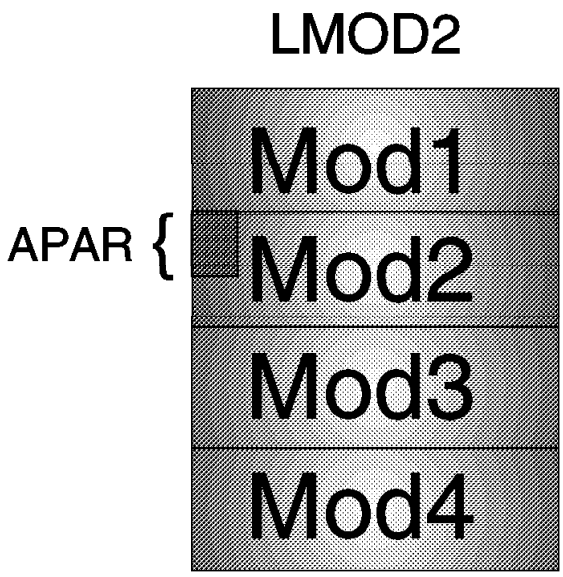


Figure 167. Fixing problems with an element (APAR)

3.2.3 Fixing problems with an element - the APAR SYSMOD

You may sometimes find it is necessary to correct a serious problem that occurs on your system before a PTF is ready for distribution. In this situation, IBM supplies you with an authorized program analysis report (APAR). An APAR is a fix designed to quickly correct a specific area of an element or replace an element in error. You install an APAR SYSMOD to implement a fix, thereby updating the incorrect element.

In the visual, the shaded section shows an area of Mod2 containing an error. The processing of the APAR SYSMOD provides a modification for object module Mod2. During the installation of the APAR SYSMOD, Mod2 is updated (and corrected) in load module LMOD2.

Figure 168 shows an example of a simple APAR SYSMOD:

```
++APAR(APAR001)             /* SYSMOD type and identifier.   */.
++VER(Z038) FMID(FUN0001)   /* Apply to this product         */
            PRE(UZ00004)    /* at this service level.        */.
++ZAP(MOD2)                 /* Update this module            */
            DISTLIB(AOSFB)  /* in this distribution library. */.
...
... zap control statements
...
```

Figure 168. Example simple APAR SYSMOD

The APAR SYSMOD always has the installation of a function SYSMOD as a prerequisite, and can also be dependent upon the installation of other PTF or APAR SYSMODs.

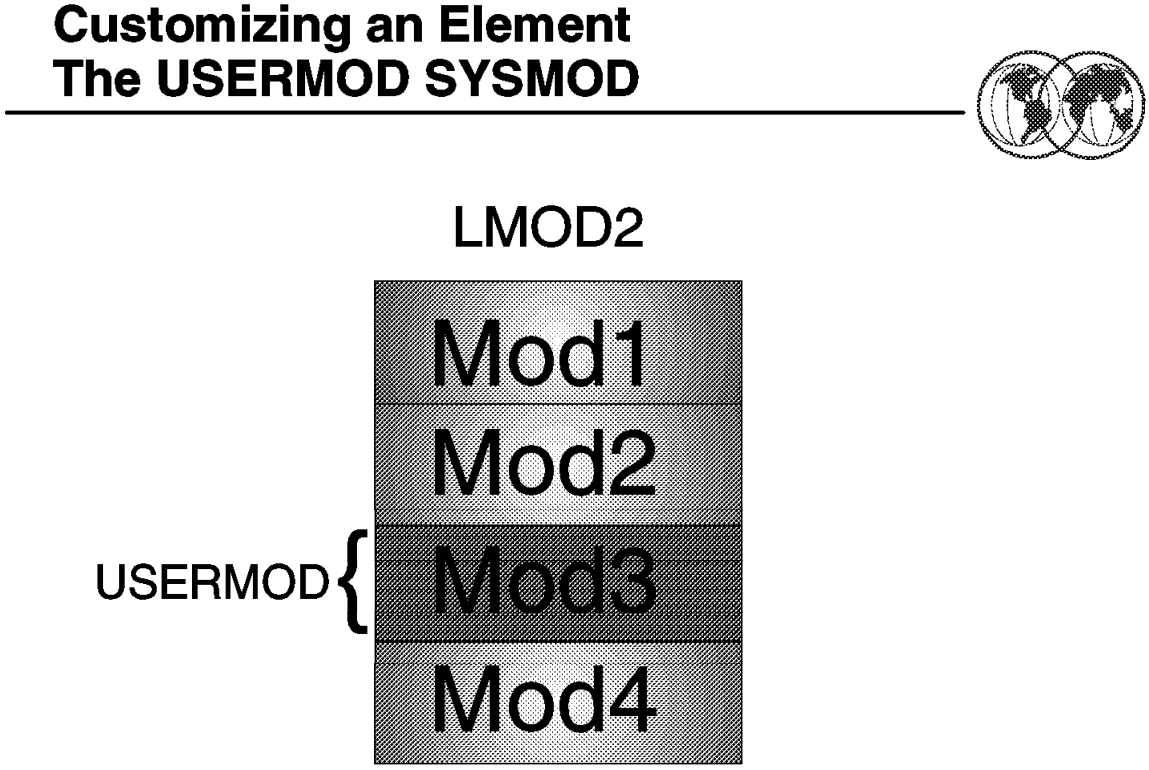


Figure 169. Customizing an element - the USERMOD SYSMOD

3.2.4 Customizing an element - the USERMOD SYSMOD

If you had a requirement for a product to perform differently from the way it was designed, you might want to customize that element of your system. IBM provides you with certain modules that allow you to tailor IBM code to meet your specific needs. After making the desired changes, you add these modules to your system by installing a USERMOD SYSMOD. This SYSMOD can be used to replace or update an element, or to introduce a totally new user-written element into the system. In either case, the USERMOD SYSMOD is built by you either to change IBM code or to add your own code to the system.

In the visual, Mod3 has been updated through the installation of an USERMOD SYSMOD.

Figure 170 shows an example of a simple USERMOD SYSMOD:

```
++USERMOD(USRMOD1)             /* SYSMOD type and identifier.   */.
++VER(Z038) FMID(FUN0001)      /* Apply to this product         */
            PRE(UZ00004)       /* at this service level.        */.
++SRCUPD(JESMOD3)              /* Update this source module     */
            DISTLIB(AOSFB)     /* in this distribution library. */.
...
... update control statements
...
```

Figure 170. Example simple USERMOD SYSMOD

Prerequisites for USERMOD SYSMODs are the installation of a function SYSMOD, and possibly the installation of other PTF, APAR, or USERMOD SYSMODs.

3.2.4.1 SYSMOD prerequisites

As you have learned, PTF, APAR, and USERMOD SYSMODs all have the function SYSMOD as a prerequisite. In addition to their dependencies on the function SYSMOD:

- PTF SYSMODs may be dependent upon other PTF SYSMODs.
- APAR SYSMODs may be dependent upon PTF SYSMODs and other APAR SYSMODs.
- USERMOD SYSMODs may be dependent upon PTF SYSMODs, APAR SYSMODs, and other USERMOD SYSMODs.

Consider the complexity of these dependencies. When you multiply that complexity by hundreds of load modules in dozens of libraries, the need for a tool like SMP/E becomes apparent.

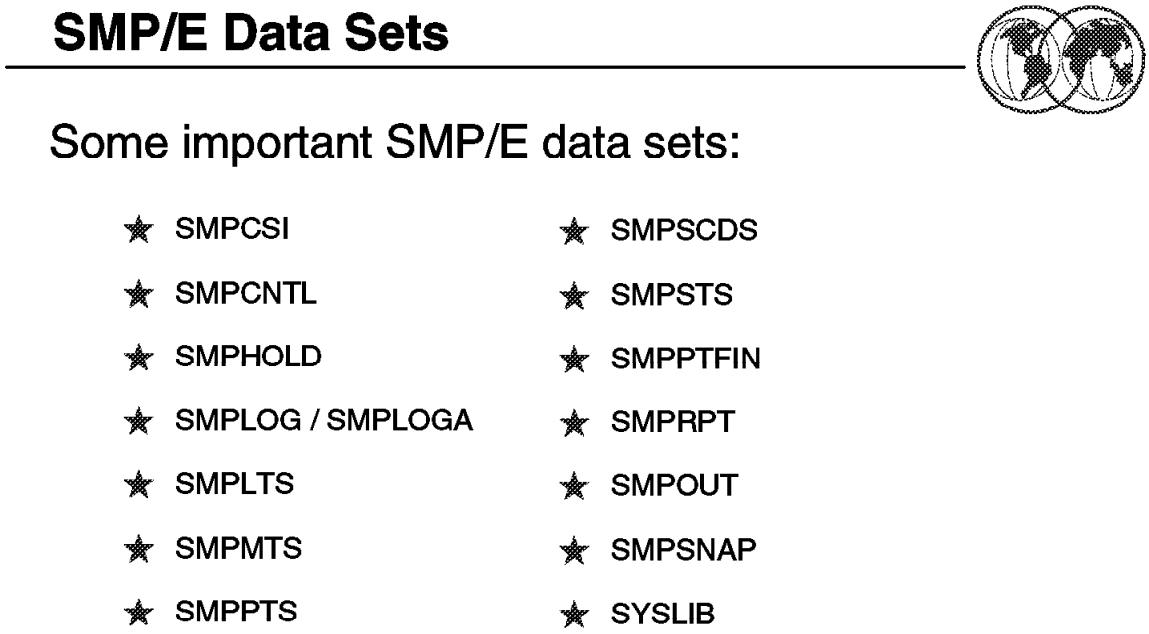


Figure 171. SMP/E data sets

3.3 Data sets used by SMP/E

When SMP/E processes SYSMODs, it installs the elements in the appropriate libraries and updates its own records of the processing it has done. SMP/E installs program elements into two types of libraries:

- *Target libraries* contain the executable code needed to run your system (for example, the libraries from which you run your production system or your test system).
- *Distribution libraries* (DLIBs) contain the master copy of each element for a system. They are used as input to the SMP/E GENERATE command or the system generation process to build target libraries for a new system. They are also used by SMP/E for backup when elements in the target libraries have to be replaced or updated.

In order for SMP/E to install elements in these libraries, it uses a database made up of several types of data sets:

SMPCSI (CSI) These data sets are Virtual Sequential Access Method (VSAM) data sets used to control the installation process and record the results of processing.

SMPPTS A data set for temporary storage of SYSMODs waiting to be installed. The PTS is used strictly as a storage data set for SYSMODs. The RECEIVE command stores SYSMODs directly on the PTS without any modifications of SMP/E information. The PTS is related to the global zone that contain information about the received SYSMODs. Only one PTS can be used for a given global zone. Therefore, you can look at the global zone and PTS as a pair of data sets to be processed.

SMPSCDS Contains backup copies of target zone entries modified during APPLY processing. Therefore, each SCDS is directly related to specific target zone, and each target zone must have its own SCDS.

SMPMTS A library in which SMP/E stores copies of macros during installation when another target macro library is identified. Therefore, the MTS is related to a specific target zone, and each target zone must have its own MTS data set.

SMPSTS A library in which SMP/E stores copies of source during installation, when another target source library is identified. Therefore, the STS is related to a specific target zone, and each target zone must have its own STS data sets.

SMPLTS A library that maintains the base version of a load module. The load module in this library specifies a SYSLIB allocation in order to implicitly include modules. Therefore, the LTS is related to a specific target zone, and each target zone must have its own LTS data set.

Other data sets used by SMP/E are known as the utility and work data sets. Some of the important utility and work data sets are:

SMPCNTL Contains the SMP/E commands to be processed.

SMPHOLD Contains ++HOLD and ++RELEASE statements to be processed by the RECEIVE command. 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).

SMPLOG 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 the data set allocation. Each zone should have its own SMPLOG data set.

SMPLOGA The backup LOG data set. If SMPLOGA is defined, it is used automatically when the SMPLOG data set is full. Each zone should have its own SMPLOGA data set.

SMPPTFIN Contains SYSMODs and ++ASSIGN statements to be processed by the RECEIVE command. It can refer to an actual data set or to a file on a tape (such as file 1 on a ESO or CUM tape).

SMPRPT Contains the reports produced during SMP/E processing. If SMPRPT is not defined, all report output goes to the SMPOUT data set.

SMPOUT 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.

SMPSNAP Used for snap dump output. When a server 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.

SYSLIB The SYSLIB is a concatenation of macro libraries that are to be used by the assembler utility.

For the APPLY and RESTORE processing, the data sets should be concatenated in this order:

1. SMPMTS
2. MACLIB
3. MODGEN
4. Target system macro libraries (such as libraries specified for SYSLIB on the ++MAC statement)

5. Distribution macro libraries (such as libraries specified for DISTLIB on the ++MAC statement).

For more information on the rest of the data sets that SMP/E uses, see *OS/390 SMP/E Reference*, SC28-1806.

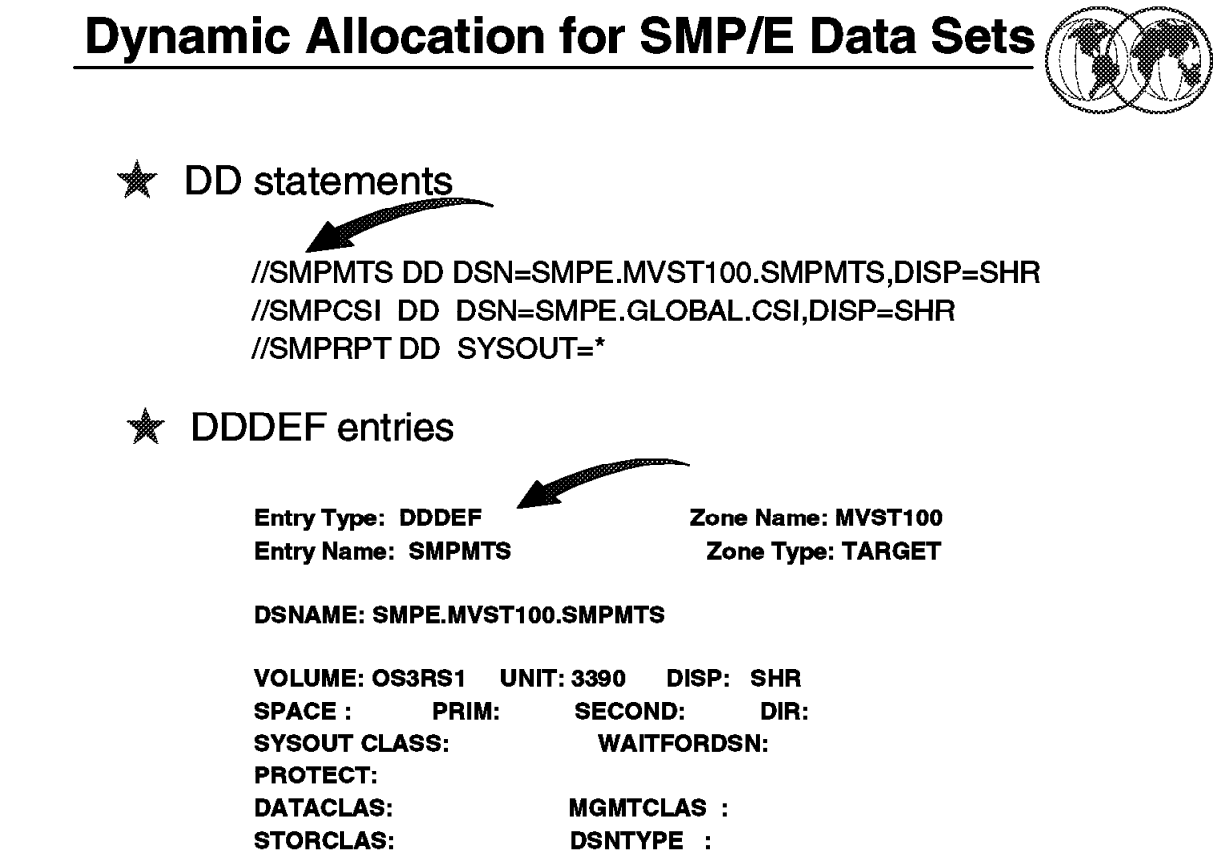


Figure 172. Dynamic allocation

3.3.1 Dynamic allocation of SMP/E data sets

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.

The advantage of using dynamic allocation is that data sets are allocated only as they are needed, compared to DD statements which require all data sets to be successfully allocated, regardless of whether they are needed for the command being processed. DDDEF entries also provide more flexibility than DD statements; they enable different zones to use different data sets for the same ddname.

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 DD statements that point to each of those zones and their associated CSIs. With the use of dynamic allocation, you eliminate all these problems.

SMP/E uses the following sources of information to allocate data sets dynamically:

- DDDEF entries
- Module GIMMPDFT
- Standard defaults

3.3.1.1 DD definition (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)

In the DDDEF entries, you can include the following information describing the data sets for dynamic allocation:

- 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 hierarchical file system (HFS)

Note: 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.

3.3.1.2 Module GIMMPDFT

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 zone in the SET command. CSECT GIMMPDFT in this module contains two tables:

- Table 1

  This table defines ddnames that may be allocated to the SYSOUT data set (for background processing) or to the terminal (for foreground processing).

- Table 2

  This table defines ddnames that can be allocated for the SMPWRKx and SYSUTx data sets.

  Note: Although you can use GIMMPDFT to define temporary data sets, it is easier to tailor the definition to your system if you use DDDEF entries.

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 that they have no effect on SMP/E dynamic allocation.

SMP/E provides a sample USERMOD (SMP0001) containing superzap statements for entries for both tables. You can find this USERMOD in the member GIMZPDFT in SYS1.SAMPLIB, which you can tailor to suit your environment or install as it is.

3.3.2 Standard defaults

The SMPOUT and SYSPRINT data sets are critical to proper SMP/E processing. 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.

For more information, see *OS/390 SMP/E Reference*, SC28-1806.

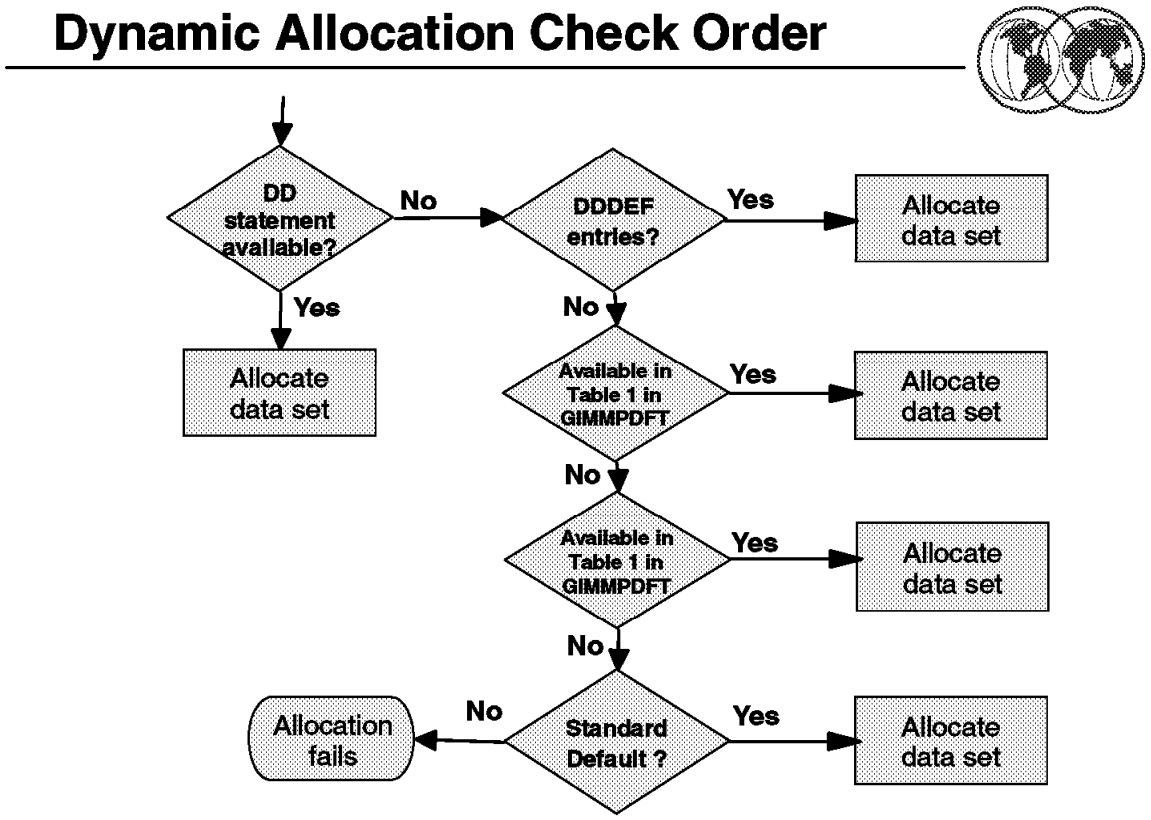


Figure 173. Dynamic allocation check sequence

3.3.3 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, the data set allocation fails.

For each data set SMP/E tries to allocate, either dynamically or through DD statements, it writes information regarding the allocation to the File Allocation report. 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

Generally, data sets that have been dynamically allocated remain allocated until the next SET command is processed. However, 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 in some way, SMP/E fails to allocate a data set dynamically or 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 allocate 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.

Notes:

1. When a data set is part of a concatenation, such as the SYSLIB concatenation, SMP/E does not go through the standard checking sequence as described before. 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.
3. 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 for each SET command. 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.

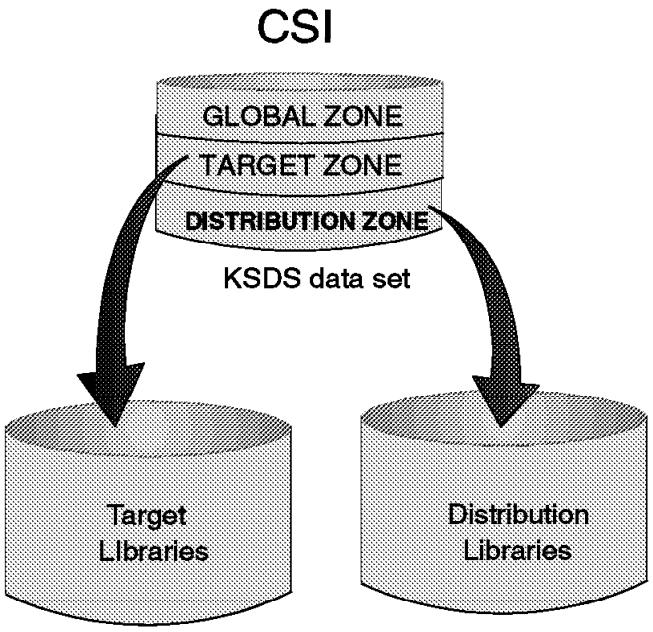


Figure 174. Consolidated Software Inventory (CSI)

3.4 Consolidated Software Inventory (CSI)

The CSI data sets contain all the information SMP/E needs to track the target and distribution libraries. The CSI contains an entry for each element in its system, which describes the element name, type, history, how the element was introduced into the system, and a pointer to the element in the target and distribution libraries. The CSI does not contain the element itself, but rather a description of the element it represents.

3.4.1 The organization of the CSI data set

In the CSI, entries for the elements in the distribution and target libraries are grouped according to their installation status. This groupings are known as SMP/E zones.

There are three types of zones in a CSI:

Global zone Contains entries needed to identify and describe each target and distribution zone to SMP/E and stores information about SMP/E processing options.

Contains status information for all SYSMODs SMP/E has begun to process and holds exception data for SYSMODs requiring special handling or that are in error.

Target zone Contains information that describes the content, structure, and status of the target libraries. It also contains a pointer to the related distribution zone, which can be used in APPLY, RESTORE, and LINK when SMP/E is processing a SYSMOD and needs to check the level of the elements in the distribution libraries.

Distribution zone Contains information that describe the content, structure, and status of the distribution libraries. Each distribution zone also points to the related target zone, which is used when SMP/E is accepting a SYSMOD and needs to check if the SYSMOD has already been applied.

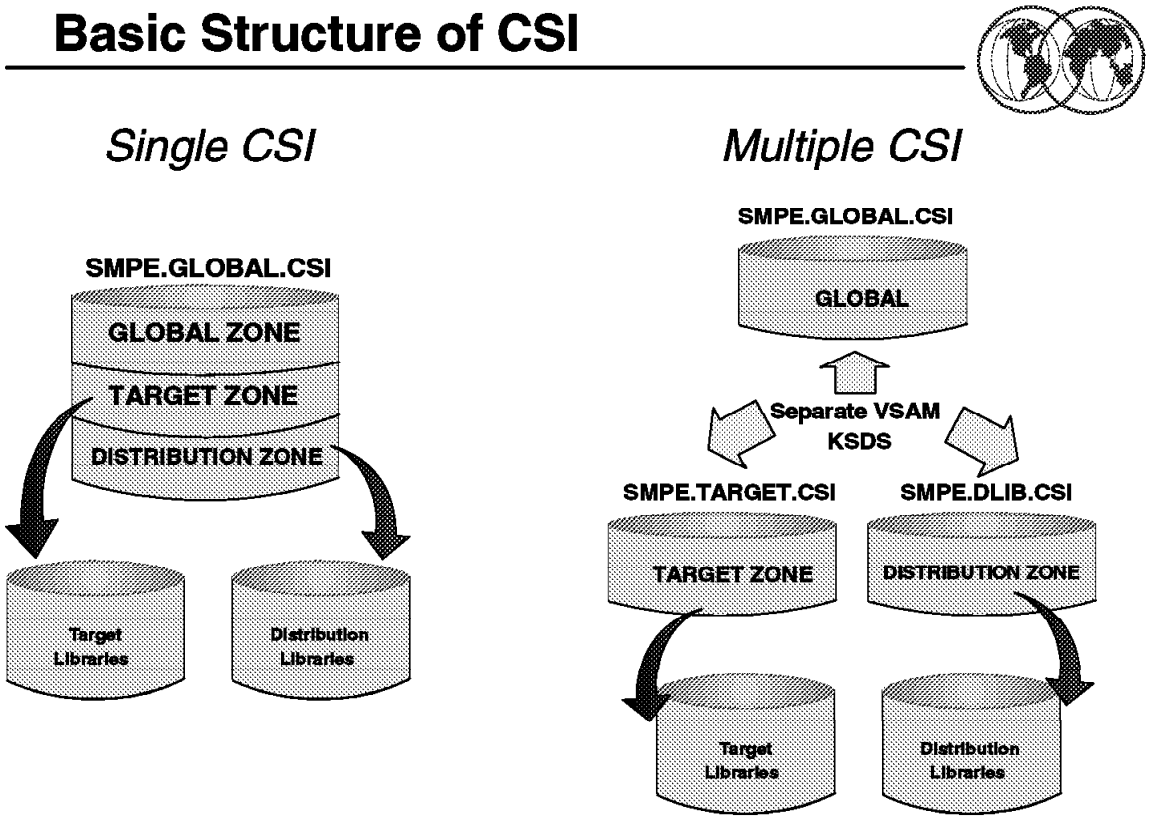


Figure 175. Basic structure of CSI

3.4.2 How to organize CSI data sets

Before you allocate any CSI data sets, you must decide how to organize those data sets. There are two basic structures for CSI data sets:

- Single CSI

  In single CSI structure, you can define one CSI to keep track of all the system activity. The single CSI data has one global zone and one or more target and distribution zones. There are some advantages of using this structure:

  – The single CSI data set optimizes the use of direct access storage.

  – The single CSI data set puts your whole establishment in one VSAM data set. This provides you with a single control point and one source of information for your whole system.

  Note: One SMP/E job can process at a time with this structure because MVS enqueues on the data set name.

- Multiple CSI

  Each zone resides in a separate VSAM data set. The target and distribution zone are connected by ZONEINDEX entries to a single global zone. The global zone must be in one of the CSI data sets.

3.4.3 How to allocate a CSI data set

After you have decided which CSI structure to adopt for your installation, you can now allocate a CSI data set using the VSAM access method services. Figure 176 on page 266 shows sample JCL for allocating a CSI data set with enough space to have multiple target and distribution zones:

```
//DEFINE  JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//STEP01   EXEC  PGM=IDCAMS
//CSIVOL   DD    UNIT=3380,VOL=SER=volid1,DISP=SHR
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
  DEFINE CLUSTER(                          -
                 NAME(SMPE.SMPCSI.CSI) 1       -
                 FREESPACE(10 5)           -
                 KEYS(24 0) 2                  -
                 RECORDSIZE(24 143)        -
                 SHAREOPTIONS(2 3) 3           -
                 UNIQUE 4                      -
        VOLUMES(volid1)            -
                )                          -
            DATA(                          -
                 NAME(SMPE.SMPCSI.CSI.DATA)  -
                 CONTROLINTERVALSIZE(4096)   -
                 CONTROLINTERVALSIZE(4096)   -
                 CYLINDERS(250 20)           -
                )                          -
           INDEX(                          -
                 NAME(SMPE.SMPCSI.CSI.INDEX) -
                 CYLINDERS(5 3)            -
                )                          -
         CATALOG(user.catalog)
/*
```

Figure 176. Sample JCL for allocating a CSI data set

1 The high level qualifier should not be SYS1 if the CSI data set is to be cataloged in a user catalog. The low level qualifier must be CSI.

2 The CSI is a key-VSAM (KSDS) data set.

3 SMP/E does not support cross-system sharing of the CSI; you cannot specify 4 as the cross-system value for SHAREOPTIONS.

4 The UNIQUE parameter ensures that the CSI cluster is assigned the name specified in the NAME parameter, rather than a VSAM-assigned name.

3.4.4 How to initialize a CSI data set

Before you can use a CSI, you must initialize it with the GIMPOOL record, which is in SYS1.MACLIB. Use the access method service shown in Figure 177 on page 267 to initialize the newly allocated CSI data set:

```
//ALLOC   JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//AMS      EXEC PGM=IDCAMS
//SMPCSI   DD   DSN=SMPE.SMPCSI.CSI,DISP=OLD
//ZPOOL    DD   DSN=SYS1.MACLIB(GIMZPOOL),DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
           REPRO OUTFILE(SMPCSI) -
           INFILE(ZPOOL)
/*
```

Figure 177. Sample job to initialize the CSI data set

Notes

1. If you use the access method services REPRO command to copy an entire CSI data set to a newly created CSI data set, you do not have to initialize your newly allocated CSI with a GIMZPOOL record. A GIMZPOOL record is copied in when the existing CSI is copied into your new CSI.
2. Make sure you use the correct SMP/E release GIMZPOOL record to initialize your CSI created with that SMP/E release. For example, use the OS/390 V2R7 SMP/E GIMZPOOL record to initialize CSI created for OS/390 V2R7 SMP/E.

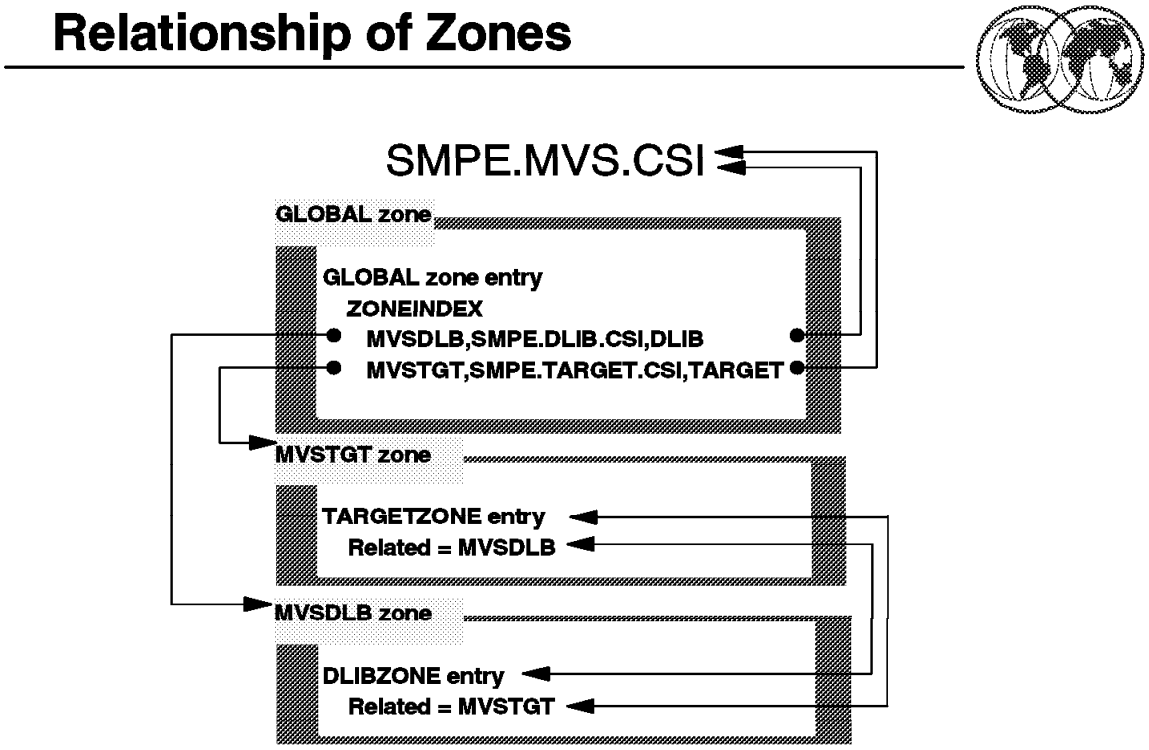


Figure 178. Relationship of zones in CSI

3.4.5 Defining zones for your system

Once you have allocated and initialized the CSI data sets, you need to create within them the entries SMP/E uses to maintain your system. The first entries you need to define are the zone definition entries:

GLOBALZONE entry A global zone is created by defining a GLOBALZONE entry. 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 in different CSI data sets. The GLOBALZONE entry must be defined before you can do other processing for that global zone.

TARGETZONE entry A target zone is created by defining a TARGETZONE entry. The TARGETZONE entry contains information SMP/E uses to process a specific target zone and the associated target libraries. It must be defined before you can do any other processing for that target zone.

DLIBZONE entry A distribution zone is created by defining a DLIBZONE entry. The DLIBZONE entry contains the information SMP/E uses to process a specific distribution zone and the associated distribution libraries. It must be defined before you can do any other processing for that distribution zone.

After you have defined the zones for your system, you can create other entries. SMP/E zones contain two basic types of entries:

- Entries controlling SMP/E processing

You generally define processing control entries through the SMP/E administration dialogs or with the UCLIN command.

- Entries describing the structure and status of the target and distribution libraries.

  Status and structure entries are generally created by SMP/E when you install SYSMODs, run the JCLIN command, or copy entries from one zone to another.

For more information about these entries, see the *OS/390 SMP/E User's Guide*, SC28-1740.

Note: SMP/E provides a member GIMSAMPU in the SYS1.SAMPLIB containing sample UCLIN statements to define entries for a basic OS/390 system. The sample definitions are syntactically correct and can be used as the basis for your CSI entries. However, this sample is not complete for all systems, but it is an example of the types of information various entries need.

For examples of using UCLIN to define entries, see *OS/390 SMP/E Commands*, SC28-1805.

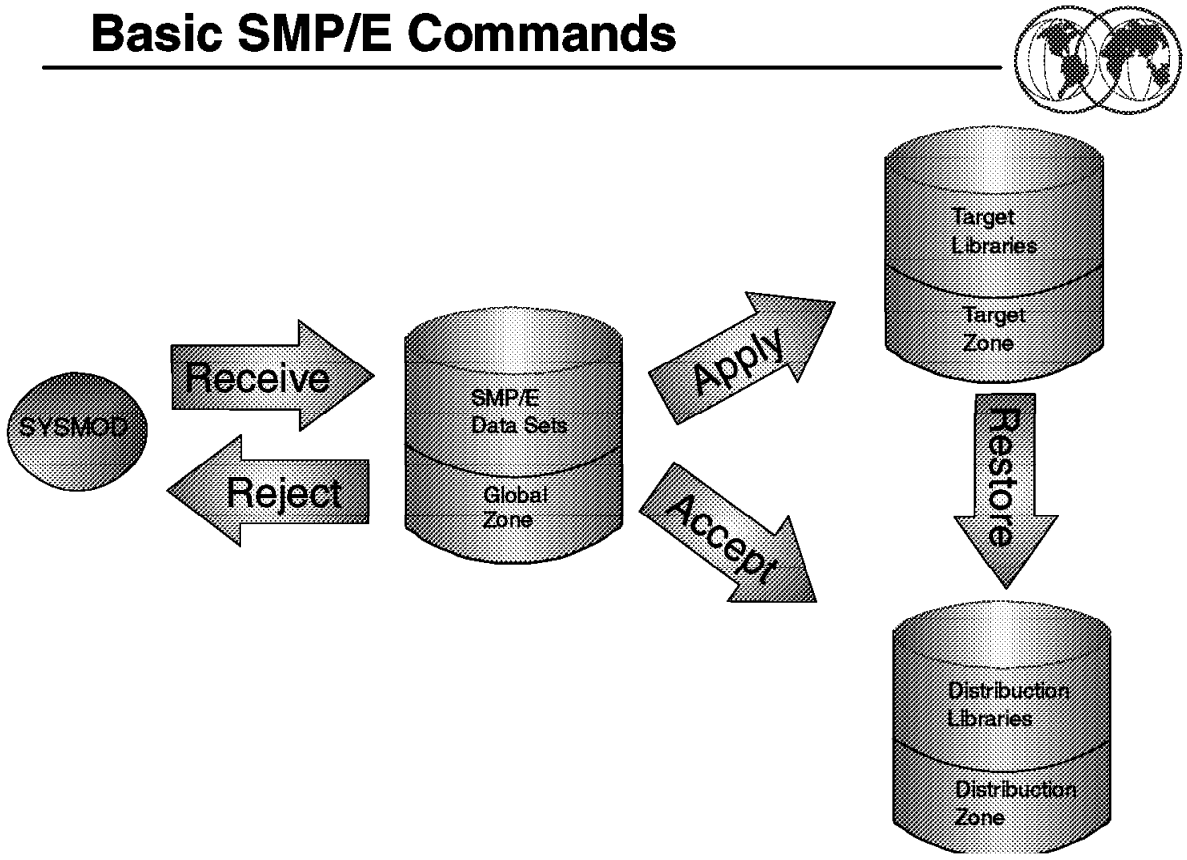


Figure 179. Basic SMP/E commands

3.5 SMP/E commands you need to know

Now that you are familiar with SMP/E and what it can do, you are probably wondering what you need to know to get started using SMP/E. Let′s take a look at the basic processing commands you need to know to use SMP/E.

3.5.1.1 Setting the zone you want to work on

Before processing SMP/E commands, you must first set the zone on which you want SMP/E to work (global, target, or distribution). You do this by issuing the SET command. The SET command identifies the zone and, therefore, the libraries, upon which subsequent SMP/E commands are to act.

The SET command can also be used to request a particular set of predefined processing options. For more information about the SET command, see *OS/390 SMP/E Commands*, SC28-1805.

3.5.1.2 Receiving the SYSMOD into SMP/E′s data set

For SMP/E to install a SYSMOD, the SYSMOD must be *received* into data sets that can be used by SMP/E. The SMP/E RECEIVE command performs the task of copying the SYSMOD from the distribution medium from which it was sent into the data sets used by SMP/E.

3.5.1.3 Applying the SYSMOD to the target libraries

After a SYSMOD has been received, you want to *apply* the SYSMOD to the appropriate target libraries. The SMP/E APPLY command invokes various system utilities to install the SYSMOD′s elements into the target libraries.

3.5.1.4 Restoring the target libraries to the previous level

Should you experience problems after applying a SYSMOD, you may want to *restore* its elements in error to a previous and stable level. The SMP/E RESTORE command replaces a failing element with a copy from the distribution libraries.

3.5.1.5 Accepting the SYSMOD and updating the distribution libraries

After you have performed a SYSMOD RECEIVE and APPLY, you want to *accept* the elements into the distribution libraries for backup. However, this should be done only after you are satisfied with the performance and stability of the elements of the SYSMOD. Once you ACCEPT a SYSMOD, you cannot RESTORE its element to a previous level. The SMP/E ACCEPT command updates the distribution libraries so they are available for backup of an future SYSMODs.

3.5.2 Displaying SMP/E data

The SMP/E CSI and other primary data sets contain a great deal of information you may find useful when installing new elements or functions preparing user modifications, or debugging problems. There are several ways SMP/E allows you to display that information, as well as information about modules, macros, and other elements:

- LIST commands
- REPORT commands
- Query dialogs

RECEIVE Processing

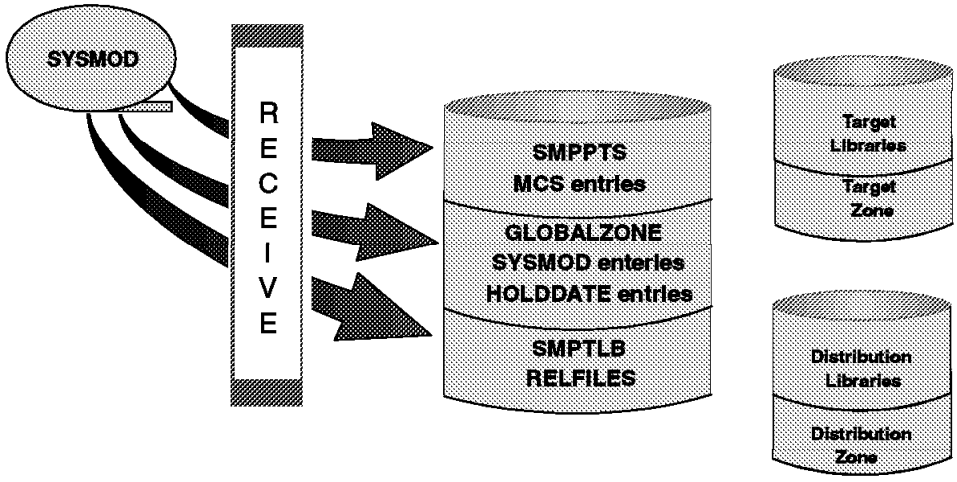


Figure 180. The RECEIVE process

3.6 Receiving SYSMODs

RECEIVE is the first SMP/E command to process any SYSMOD. You use the RECEIVE command to load the SYSMOD information from the distribution medium into the SMPPTS and SMPTLIB data sets for later installation of the SYSMODs.

As mentioned in the previous section, each SYSMOD processed by SMP/E contains two types of information:

- Instructions telling SMP/E what elements are in the SYSMOD and how to install them.
- The actual element replacements or updates contained in the SYSMOD.

3.6.1 Packaging of the SYSMODs

The element replacements or updates in the SYSMODs can be packaged in the following ways:

RELFILE This method packages the elements in relative files that are separate from the modification control statements (MCSs). This is the most frequently used method for function SYSMODs.

INLINE This method packages the elements immediately following the associated MCSs.

INDIRECT LIBRARY This method packages elements in DASD data sets that are separate from the MCSs.

For more details about packaging, see *Standard Packaging Rules for MVS-Based Products*, SC23-3695.

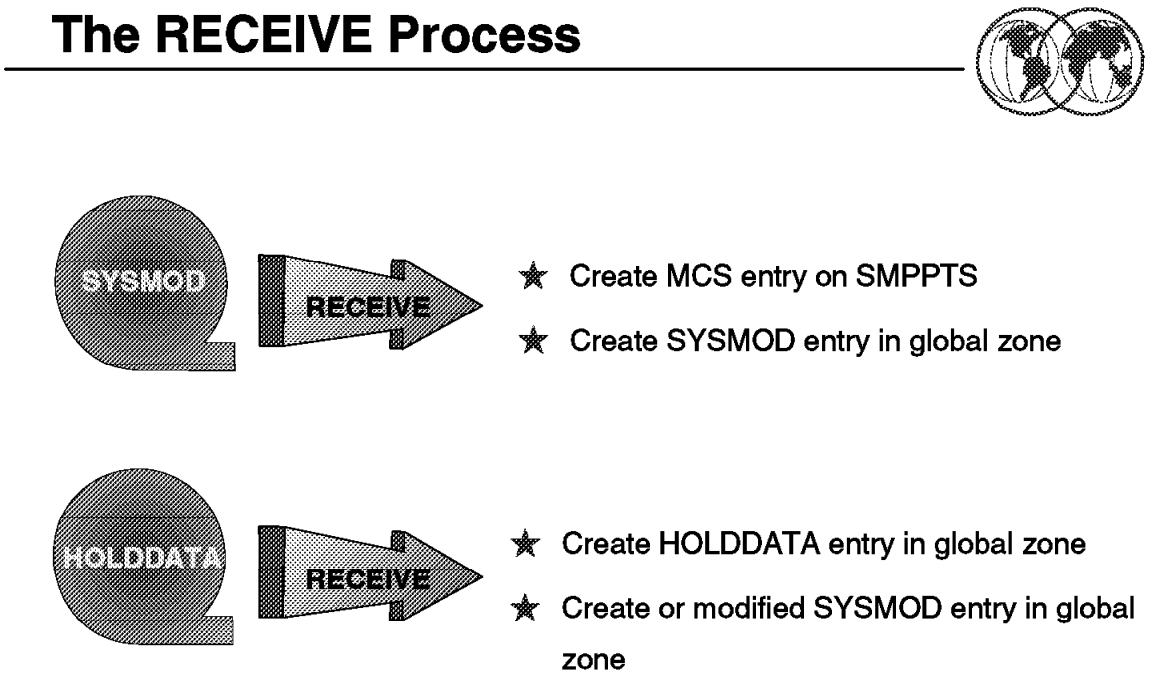


Figure 181. The RECEIVE process

3.6.2 The RECEIVE Process

During RECEIVE processing, SMP/E reads data from tape files or DASD data sets into the global zone, the SMPPTS, and temporary data (SMPTLIBs) for later 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, and ++PRODUCT statements.
- The *SMPHOLD* data sets, which contains exception SYSMOD data (++HOLD and ++RELEASE statements).

An example of SMPTLIB data sets is shown in Figure 182 on page 275.

```
 Menu  Options  View  Utilities  Compilers  Help
------------------------------------------------------------------------
DSLIST - Data Sets Matching SYS1.MVS                        Row 1 of 4
Command ===>                                              Scroll ===> PAGE

Command - Enter "/" to select action            Message          Volume
------------------------------------------------------------------------
        SYS1.MVS.JTCP356.F1                                      MPCAT1
        SYS1.MVS.JTCP356.F2                                      MPCAT1
        SYS1.MVS.JTCP356.F3                                      MPCAT1
        SYS1.MVS.JTCP356.F4                                      MPCAT1
*************************** End of Data Set list ********************
```

Figure 182. Example SMPTLIB data sets

When receiving a SYSMOD, SMP/E crates two entries:

1. An MCS entry is created on the SMPPTS. This entry is an exact copy of the SYSMOD as it appeared in the SMPPTFIN data set.
2. A SYSMOD entry is created in the global zone. This entry contains information that describes the installation requirements and element content of the SYSMOD.

When receiving the HOLDDATA, SMP/E also creates (or modifies) two entries:

1. A HOLDDATA entry is created (or modified) in the global zone. This entry is an exact copy of the ++HOLD statements as they appeared in the SMPHOLD data set. The name of the entry is the ID of the SYSMOD affected by this ++HOLD statement. The HOLDDATA entry for a single SYSMOD can contain multiple ++HOLD statements.

   Note: When a ++RELEASE statement is processed, SMP/E removes the corresponding ++HOLD statement from the HOLDDATA entry. When all ++HOLD are removed, the HOLDDATA entry is automatically deleted.

2. A SYSMOD entry is created (or modified) in the global zone. This entry contains information that describes the exception SYSMOD conditions.

   For each ++HOLD statement processed, SMP/E updates the global zone SYSMOD entry to add a HOLD reason ID subentry. There are three types of HOLD reason ID subentries, HOLDERROR, HOLDSYSTEM, and HOLDUSER, corresponding to the three categories of exception SYSMODs.

   Note: When a ++RELEASE statement is processed, SMP/E removes the corresponding reason ID from the global zone SYSMOD entry.

3.6.3 Managing exception SYSMOD through HOLLDATA

In SMP/E, when we speak of exception data, we are usually referring to HOLDDATA. HOLDDATA is often supplied for a product to indicate a specific SYSMOD should be held from installation. Reasons for holding a SYSMOD can be:

- A PTF is in error, normally known as PE, and should not be installed until the error is corrected (ERROR HOLD).
- Certain system actions may be required before SYSMOD installation (SYSTEM HOLD).
- The user may want to perform some actions before installing the SYSMOD (USER HOLD).

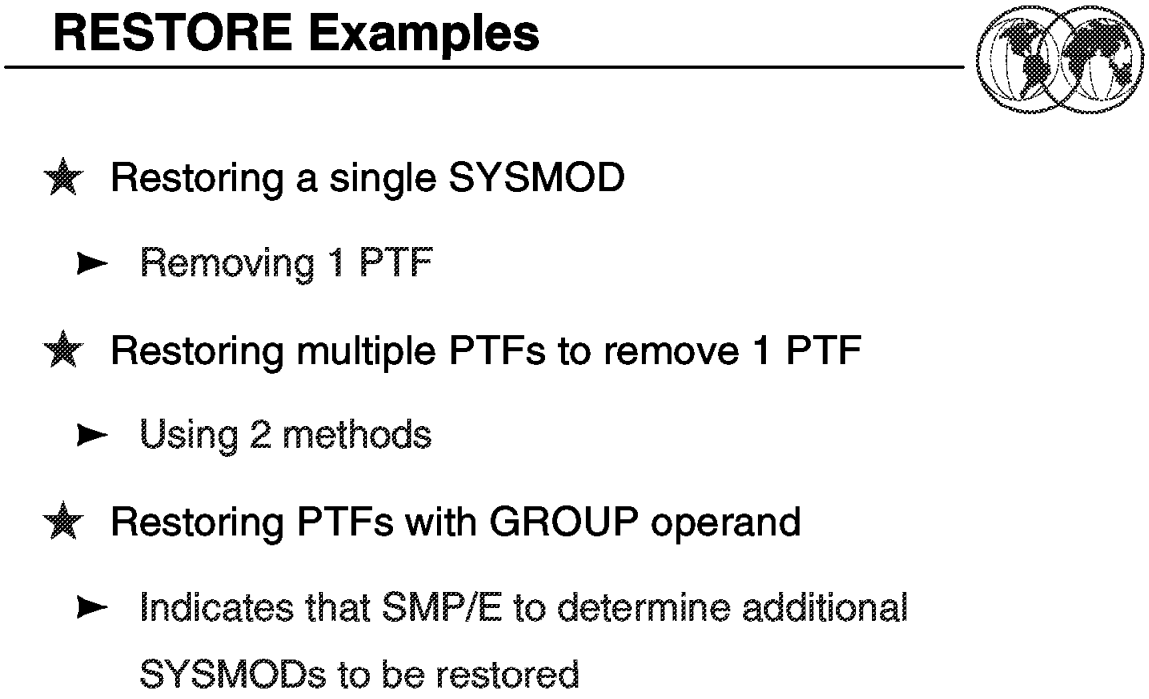


Figure 183. Sources of HOLDDATA

3.6.3.1 Sources of HOLDDATA

The main sources of HOLDDATA provided by IBM are:

- *Custom Build Product Delivery Option (CBPDO)* tapes

  This contains HOLDDATA that has been customized to your product set. That is, it contains only data applicable to PTFs for those products within a given feature that you have ordered.

- *Expanded Service Option (ESO)* tapes

  IBM regularly creates the service levels shipped on these tapes, then custom-builds ESOs for users and makes the tapes available through either subscription orders or special request orders.

- *Cumulative Service (CUM)* tapes

  This tape is sent together with an order for a new function that contains all the current PTFs and also the HOLDDATA applicable to those PTFs.

- *Preventive service planning (PSP) information* from the CSSF files

  Once a service level has been created, there is no further opportunity to change the HOLDDATA on that tape, even though new errors are reported. PSP files have been set up to hold this additional HOLDDATA.

3.6.4 SMP/E data sets used in the RECEIVE Process

The following data sets may be needed to run the RECEIVE command. You can defined them using DD statements in your job or, normally, by DDDEF entries:

SMPCNTL	SMPLOGA	SMPRPT	SYSUT1	zone
SMPCSI	SMPOUT	SMPSNAP	SYSUT2	
SMPHOLD	SMPPTFIN	SMPTLIB	SYSUT3	
SMPLOG	SMPPTS	SMPPRINT		

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 regarding those SYSMODs received.
2. You only required the SMPHOLD DD statement if exception SYSMOD data is to be received from the SMPHOLD data set.
3. You are required to have SMPPTFIN DD statement if SYSMODs or ++ASSIGN statements are to be received from the SMPPTFIN data set.
4. The *zone* represents the DD statement required for each target or distribution zone used by the command. If no DD statement is specified, SMP/E dynamically allocates data sets using the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements can be used to override the ZONEINDEX information, they are not a substitute for a ZONEINDEX. A ZONEINDEX is always required for a zone.

For a detailed description of these data sets, see *OS/390 SMP/E Reference*, SC28-1806.

RECEIVE Examples

- ★ Receiving only HOLDDATA
- ★ Receiving only SYSMOD
- ★ Receiving both SYSMOD and HOLDDATA
- ★ Receiving selected SYSMOD and HOLDDATA

Figure 184. RECEIVE examples

3.6.4.1 RECEIVE examples

The following examples are provided to help you use the RECEIVE command.

3.6.4.2 How to receive only HOLDDATA

There may be times when you do not want to receive the SYSMODs from a service tape, but you do want to receive the HOLDDATA. Because the HOLDDATA provides information about SYSMODs requiring special handling or that are in error, it is important for you to receive the HOLDDATA into SMP/E′s storage repository as soon as possible. Figure 185 on page 279 shows sample JCL to process only the HOLDDATA:

```
//RECHOLD JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//SMPE     EXEC PGM=GIMSMP 1
//SMPCSI   DD DSN=SMPE.GLOBAL.CSI,DISP=SHR
//SMPHOLD  DD DISP=SHR,DSN=SYS1.OS251140.HOLDATA, 2
//            VOL=SER=S1140C,
//            LABEL=(1,SL),UNIT=3480
//SMPRPT   DD SYSOUT=*
//SMPOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SMPCNTL  DD *
  SET BDY (GLOBAL) . 3
  RECEIVE HOLDDATA . 4
```

Figure 185. Sample JCL to process only HOLDDATA

1 This is the SMP/E program.

2 In the SMPHOLD DD statement, you specify the data set that contains the HOLDDATA. In this case, the HOLDDATA comes with the CBPDO tape.

3 For the RECEIVE command, the SET BOUNDARY command must specify the global zone.

4 The HOLDDATA parameter indicates that the applicable data form SMPHOLD DD should be received. In this case, HOLDDATA is received for all FMIDs defined in the GLOBAL zone.

3.6.4.3 How to receive only SYSMODs

This example assumes that you have previously received the HOLDDATA from a service tape and are now ready to install the SYSMODs. Before you can install these SYSMODs (using the APPLY and ACCEPT commands), you must first receive them. Figure 186 shows sample JCL to process only the SYSMODs:

```
//RECSYSM JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//SMPE     EXEC PGM=GIMSMP
//SMPPTFIN DD DISP=SHR,DSN=SYS1.OS251140.PTF, 1
//            VOL=SER=S1140C,
//            LABEL=(2,SL),UNIT=3480
//SMPCNTL  DD *
  SET BDY (GLOBAL) .
  RECEIVE SYSMODS SOURCEID(MVSPUT1) . 2
```

Figure 186. Sample JCL to process only the SYSMODs

1 SMPPTFIN DD statement indicates the data sets which contains MCSs defining the SYSMODs to be received.

2 SYSMODS parameter indicates that only the data from SMPPTFIN should be received. SOURCEID specifies a one-to-eight 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.

3.6.4.4 How to receive both SYSMOD and HOLDDATA

In the course of maintaining your system, you need to install both the SYSMODs and process the related HOLDDATA. You can accomplish this by using the sample JCL shown in Figure 187 on page 280:

```
//RECPTF1 JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//SMPE     EXEC PGM=GIMSMP
//SMPPTFIN DD DISP=SHR,DSN=SYS1.OS251140.PTF,
//            VOL=SER=S114OC,
//            LABEL=(2,SL),UNIT=3480
//SMPHOLD  DD DISP=SHR,DSN=SYS1.OS251140.HOLDATA,
//            VOL=SER=MPWRK1
//SMPCNTL  DD *
  SET BDY (GLOBAL) .
  RECEIVE . 1
```

Figure 187. Sample JCL to receive both the SYSMODs and HOLDDATA

1 This will receive all the SYSMODs and the related HOLDDATA for all the FMIDs that were defined in the global zone.

3.6.4.5 How to receive selected SYSMODs and HOLDDATA

The RECEIVE command also allows you to select individual SYSMODs or exception HOLDDATA applicable to the selected SYSMODs. In the example shown in Figure 188, the commands caused SMP/E to receive two SYSMODs specified plus any HOLDDATA entries that are applicable to the SYSMODs:

```
//RECPTF2 JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//SMPE     EXEC PGM=GIMSMP
//SMPPTFIN DD DISP=SHR,DSN=SYS1.OS251140.PTF,
//            VOL=SER=S114OC,
//            LABEL=(2,SL),UNIT=3480
//SMPHOLD  DD DISP=SHR,DSN=SYS1.OS251140.HOLDATA,
//            VOL=SER=MPWRK1
//SMPCNTL  DD *
  SET BDY (GLOBAL) .
  RECEIVE S(UW41970,UW41979) 1
          SYSMODS
          HOLDDATA
          SOURCEID(MVSPUT1) .
/*
```

Figure 188. Sample JCL to receive selected SYSMODs and HOLDDATA

1 Only two SYSMODs and applicable HOLDDATA were received using this command.

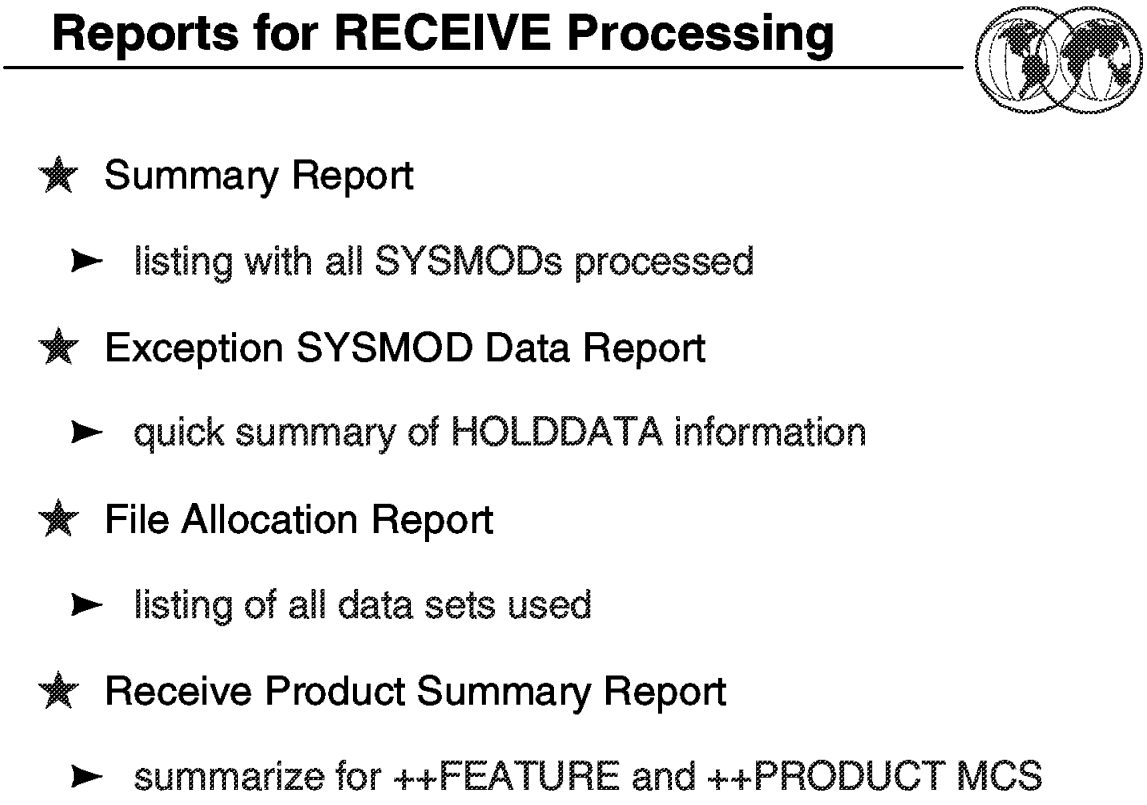


Figure 189. Reports for RECEIVE processing

3.6.5 Reports for RECEIVE processing

When RECEIVE processing is complete, the following reports will help you to analyze the results:

- *Summary Report* provides you with a listing with all the SYSMODs that were processed during the RECEIVE command run. It shows you which SYSMODs were received, which were not received, and why they were not received.

  An example of RECEIVE Summary Report is shown in Figure 190 on page 282.

```
PAGE 0001 - NOW SET TO GLOBAL ZONE  DATE 04/16/99  TIME 13:37:0 GIMSMP

  RECEIVE  SYSMODS.
PAGE 0002 - NOW SET TO GLOBAL ZONE  DATE 04/16/99  TIME 14:08:0 GIMSMP

                        RECEIVE  SUMMARY  REPORT


SYSMOD   STATUS        TYPE      SOURCEID  STATUS FIELD COMMENTS


UQ01266  NOT RECEIVED  PTF [1]       NO APPLICABLE ++VER
         NOT ASSIGNED            PUT9710   SYSMOD NOT IN RECEIVE STATUS
UQ09543  NOT RECEIVED  PTF [2]       ALREADY RECEIVED
         ASSIGNED                PUT9710
UQ09340  RECEIVED      PTF [3]
         ASSIGNED                PUT9710
```

Figure 190. Sample RECEIVE Summary Report

1 UQ01266 was not received because there is no applicable subsystem or system release (srel) defined in the GLOBALZONE that matches the ++VER statement.

2 UQ09543 was not received because it has already been received before.

3 UQ09340 was successfully received.

- *Exception SYSMOD Data Report* provides you with a quick summary of the HOLDDATA information processed during the RECEIVE command run. It lists the SYSMODs requiring special handling or that are in error, and those SYSMODs no longer requiring special handling or that have had an error fixed.

  An example of Exception SYSMOD Data Report is shown in Figure 191.

```
PAGE 0106 - NOW SET TO GLOBAL ZONE  DATE 04/16/99  TIME 14:08:02  GIMSMP

               RECEIVE ++HOLD/++RELEASE SUMMARY

NOTE:  SMD NF   - SYSMOD NOT RELEASED - NOT FOUND IN THE 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

UQ09660 SYS  HELD   ACTION HDB4410   ++ HOLD(UQ09660) SYS FMID(HDB4410)
                    [1]       ++ HOLD(UQ09660) SYS FMID(HDB4410)
                                        COMMENT
                                         (***Action for PN04918)
```

Figure 191. Sample RECEIVE Exception Data Report

1 UQ09660 is an exception SYSMOD. Normally, for exception SYSMOD with the reason ID as ACTION, you have to perform certain actions before you can install the SYSMOD. For example, you may have to IPL the system for the fix to take effect.

- *File Allocation Report* provides you with a list of data sets used for RECEIVE processing and supplies information about these data sets.

An example of the File Allocation Report is shown in Figure 192 on page 283.

```
PAGE 0106 - NOW SET TO GLOBAL ZONE  DATE 04/16/99  TIME 14:08:02  GIMSMP

                        SMP RECEIVE  FILE ALLOCATION REPORT

ZONE DDNAME DDDEFNAM SMPDDNAM TYPE  -----------DATA SET OR PATH---------

     SMPCNTL                  PERM  IBMSSAA.SMPE.MOF2.OS390.JCL
     SMPCSI                   PERM  DBAP.DB2V410.SMPG.CSI
     SMPLOG                   SYSIO IBMSSAA.IBMSSAA1.JOB29770.D0000129
     SMPLOGA                  SYSIO IBMSSAA.IBMSSAA1.JOB29770.D0000130
     SMPOUT                   SYSIO IBMSSAA.IBMSSAA1.JOB29770.D0000127
     SMPPTFIN                 PERM  SMPPTFIN
     SMPPTS   SMPPTS 1 PERM  DBAP.DB2V410.SMPPTS
     SMPRPT                   SYSIO IBMSSAA.IBMSSAA1.JOB29770.D0000128
     SYSUT1   SYSUT1          PERM  SYS99106.T133706.RA000.IBMSSAA1
     SYSUT2   SYSUT2          PERM  SYS99106.T133706.RA000.IBMSSAA1
     SYSUT3   SYSUT3          PERM  SYS99106.T133706.RA000.IBMSSAA1
```

Figure 192. Sample RECEIVE File Allocation Report

1 This shows that SMP/E dynamically allocates the SMPPTS data set through DDDEF.

- *Receive Product Summary Report* is produced at the completion of RECEIVE processing to summarize the processing that occurred for ++FEATURE and ++PRODUCT MCS.

  Note: If no ++FEATURE and ++PRODUCT MCS are processed, this report is not generated.

  An example of the RECEIVE Product Summary Report is shown in Figure 193.

```
  ++PRODUCT(5647-A01,2.5.0) DESCRIPTION(OS/390)
    SREL(Z038).
  ++FEATURE(OS3250BA) DESCRIPTION(OS/390 Base)
    PRODUCT(5647-A01,2.5.0)
    FMID(HBB6605,HMP1B00).
  ++FEATURE(OS3250DD) DESCRIPTION(OpenEdition DCE User Privacy DES)
    PRODUCT(5647-A01,2.5.0)
    FMID(JMB3125).
  ++FEATURE(OS3250LD) DESCRIPTION(Language Environment Decryption)
    PRODUCT(5647-A01,2.5.0)
    FMID(JMWL755).
```

Figure 193. Sample RECEIVE Product Summary Report

REJECT Processing

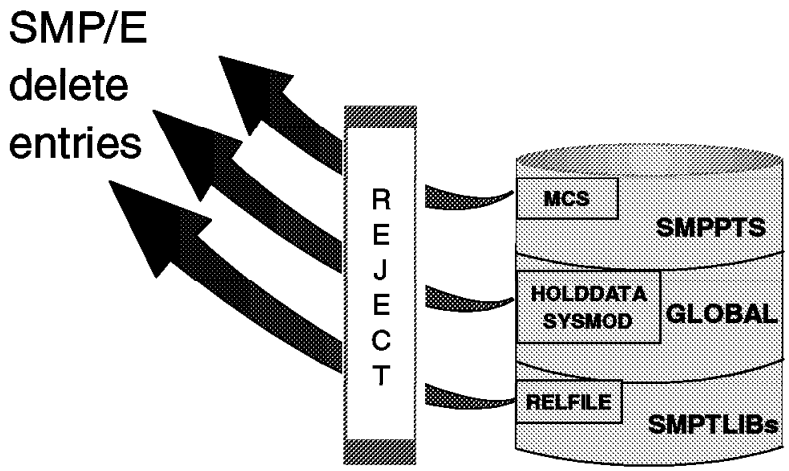


Figure 194. The REJECT process

3.7 Rejecting SYSMODs

There are times when you need to reverse the process of receiving SYSMODs into your system. After receiving a SYSMOD, you can use the REJECT command to remove that SYSMOD from the PTS and the global zone.

The REJECT processing deletes a SYSMOD's global zone entry, HOLDDATA entry (if any), PTF MCS entry, and also scratches any TLIB data sets allocated for the SYSMOD.

3.7.1 Processing modes of the REJECT command

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.

Note: The processing modes are mutually exclusive. No two modes can be used together.

There are four modes of REJECT processing:

- *MASS mode*

  SMP/E rejects all SYSMODs that have been received but not installed.

- *SELECT mode*

  SMP/E rejects specific SYSMODs that have been received but not applied.

- *PURGE mode*

SMP/E rejects all SYSMODs that have been accepted into the specified distribution zones. This is used when SYSMODs were not automatically deleted once they have been accepted into the distribution libraries. This is true when 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. This can be used to delete service for all functions that have been deleted from the global zone. It can also be used to delete FEATURE and PRODUCT entries for all functions that have been deleted from the global zone.

3.7.1.1 Entries deleted by the REJECT command

Regardless of whichever REJECT processing mode you select, SMP/E will delete the following:

- The SMPPTS MCS entry
- The global zone SYSMOD entry
- The associated FMID subentry in the GLOBALZONE entry, as appropriate
- The eligible HOLDDATA entries
- The associated SMPTLIB data sets, if the SYSMOD was packaged in RELFILE format

REJECT Examples

- Rejecting in MASS mode
- Rejecting in SELECT mode
- Rejecting in PURGE mode
- Rejecting in NOFMID mode

Figure 195. REJECT examples

3.7.1.2 REJECT examples

The following examples are provided to help you to use the REJECT command.

3.7.1.3 How to reject in MASS mode

In the MASS mode, you can reject all SYSMODs that have been received but not installed yet. You can use the sample JCL shown in Figure 196 on page 287 to perform this task:

```
//REJSYSM JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//SMPE     EXEC PGM=GIMSMP
//SMPCSI   DD DSN=SMPE.GLOBAL.CSI,DISP=SHR
//SMPRPT   DD SYSOUT=*
//SMPOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SMPCNTL  DD *
    SET      BDY(GLOBAL)        /* Set to global zone.      */.
    REJECT   APARS              /* Reject all received-only */
             FUNCTIONS          /* SYSMODs.                 */
             PTFS
             USERMODS.
/*
```

Figure 196. Sample JCL for rejecting SYSMODs in MASS mode

Note: This method may reject SYSMODs you wanted to keep, and if that happens you would have to receive them again.

Using the MASS mode, you can also reject only PTFs that have been received but not applied. The commands shown in Figure 197 shows how it can be done.

```
SET      BDY(GLOBAL) .      /* Set to global zone.      */
REJECT .                    /* Reject all received-only
                               PTFs.                    */
```

Figure 197. Rejecting PTFs that have been received but not applied

Note: If no SYSMOD types are specified on a REJECT command for MASS mode, SMP/E rejects only PTFs.

3.7.1.4 How to reject in SELECT mode

Sometimes, you want to reject certain SYSMODs from your system. You can use the REJECT command in the SELECT mode if you have applied and accepted a USERMOD into your system. You want to reject the current version, update the SYSMOD, and then reapply and reaccept it. You can use the command shown in Figure 198 to perform this task.

```
SET      BDY(GLOBAL) .        /* Set to global zone.      */
REJECT   S(MYMOD01) 1  /* Reject this SYSMOD       */
         BYPASS(              /* even though it was       */
                ACCEPTCHECK   /* accepted and             */
                APPLYCHECK) . /* applied.                 */
```

Figure 198. Rejecting a SYSMOD in SELECT mode

1 MYMOD01 is a USERMOD.

Note: By default, the REJECT command will only reject SYSMODs that are received but not installed. However, you can specify the BYPASS operand in the REJECT command to prevent SMP/E from checking where the SYSMODs have been installed.

You can also use the REJECT command to reject HOLDDATA from your system. Assuming that you want to delete a HOLDDATA entry with no associated SYSMOD entry, you can use the command shown in Figure 199 on page 288 to accomplish this task.

```
SET      BDY(GLOBAL) .      /* Set to global zone.      */
REJECT   S(UW04356)         /* For this SYSMOD,         */
         HOLDDATA .         /* reject the HOLDDATA.     */
```

Figure 199. Rejecting HOLDDATA from the system

Note: The commands will delete both the SYSMOD and its associated HOLDDATA entry if it exists in the system.

3.7.1.5 How to reject in PURGE mode

When you have specified NOPURGE in the OPTIONS entry, SMP/E will not delete the GLOBAL SYSMOD and the SMPPTS MCS entry after you have accepted the SYSMODs. You can use the commands shown in Figure 200 to reject SYSMODs that have been accepted into your system.

```
SET      BDY(GLOBAL) .      /* Set to global zone.      */
REJECT   PURGE(DLIB1) .     /* Reject SYSMODs installed
                               in this DLIB zone        */
```

Figure 200. Rejecting in PURGE mode

3.7.1.6 How to reject in NOFMID mode

There are situations where you have received, applied, and accepted function HMX1101, which is deleted from the global zone and SMPPTS once it was accepted. You have also received service for the function. Now, you want to install an updated version of the function, therefore you want to delete the FMID of the current function from the GLOBALZONE entry, together with the service and its associated HOLDDATA. You can use the command shown in Figure 201 to perform the delete.

```
 SET    BDY(GLOBAL)  .     /* Process global zone.    */
 REJECT DFMID(HMX1101)     /* Delete FMID and reject  */
        NOFMID       .     /* for FMIDs not in GZONE. */
```

Figure 201. Rejecting in NOFMID mode

Note: This command deletes the SYSMODs and HOLDDATA for FMID HMX1101 plus all functions that are not defined in the GLOBALZONE entry.

3.7.1.7 Reports for REJECT processing

At the end of REJECT processing, output from the REJECT command includes reports, as well as statistics written to SMPOUT and SMPLOG.

Two reports are produced during REJECT processing:

- File Allocation report

  This provides a list of data sets allocated during the REJECT process and also information regarding these data sets.

- REJECT Summary report

  This report is produced at the completion of REJECT processing to summarize the processing that occurred for SYSMODs and other data.

For more information regarding these reports, see *OS/390 SMP/E Commands*, SC28-1805.

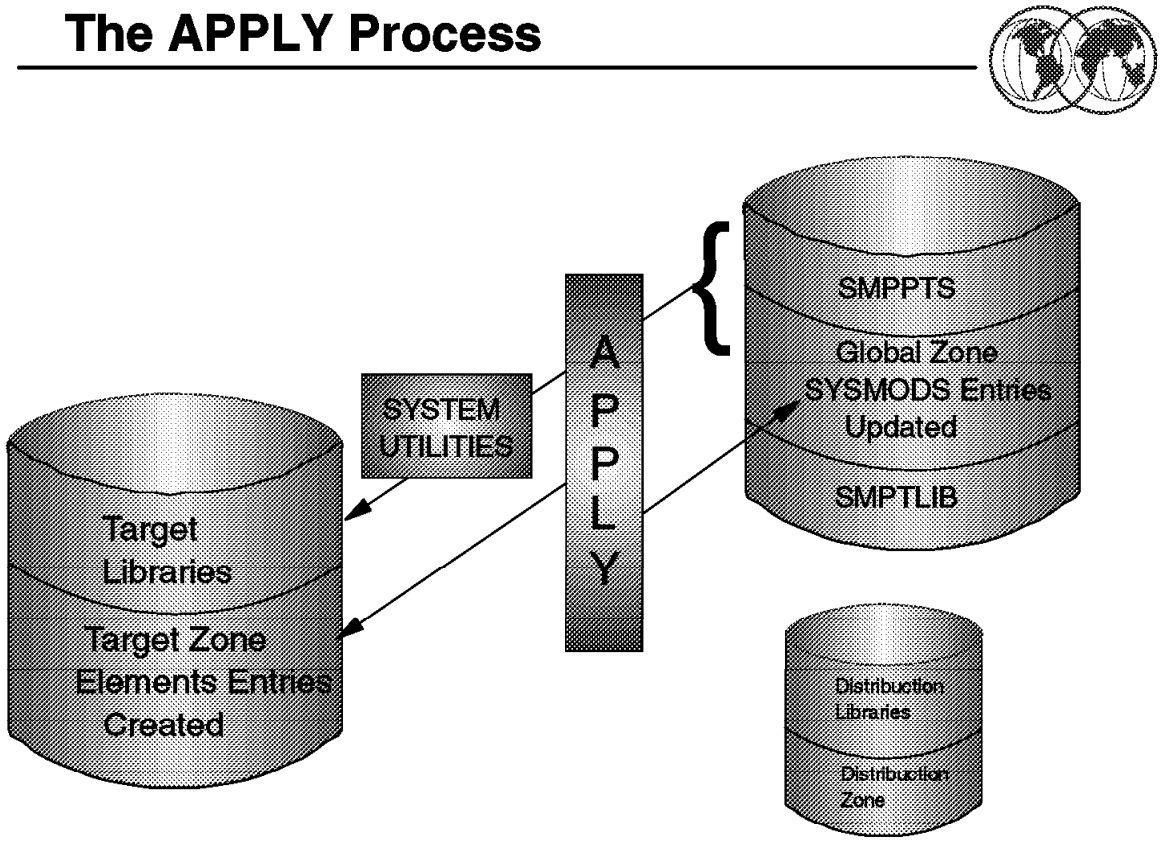


Figure 202. The APPLY process

3.8 The APPLY Process

After the SYSMODs have been received, you can use the APPL command to install them into the appropriate target system libraries. The APPLY command calls system utilities, which are responsible for the actual updating of those libraries.

3.8.1 Selecting SYSMODS

You can specify operands on the APPLY command that tell SMP/E which of the received SYSMODs are to be selected for installation in the target libraries. SMP/E checks to make sure all other required SYSMODs (prerequisites) have been installed or are being installed concurrently and in the proper sequence.

3.8.1.1 Selecting elements

During APPLY processing, SMP/E uses the information provided in the selected SYSMODs to determine which elements should be installed in the target libraries. The selection of elements is monitored by SMP/E to make sure that the correct functional level of each element is selected.

3.8.1.2 Updating the target library

After the proper SYSMODs have been selected and the proper functional and service level of each element has been determined, the APPLY command directs SMP/E to call the system utilities. It is the system utilities that actually place the elements into the target libraries described in the target zone. The source of the elements is the SMPTLIB data sets, the SMPPTS data set, or the indirect libraries, depending on how the SYSMOD was packaged.

Notes:

1. Because the APPLY command updates the system libraries, you should never use it on a live production system. When you process the APPLY command, you should always use a copy of the target libraries and target zone. By using a copy, you minimize the risk of new code causing an outage of your system. This process of copying is called *cloning* and is explained in detail in the *OS/390 Software Management Cookbook*, SG24-4775.
2. We recommend you always use the CHECK subparameter before you update the libraries with the new version of the modules, even when applying in a test system. It can avoid unnecessary workload.

3.8.2 How SMP/E keeps track of APPLY processing

SMP/E updates the information about the SYSMODs that have been applied. Remember, the target zone reflects the contents of the target libraries. Therefore, after the utility work is complete, and the target libraries have been updated, the target zone is updated to accurately reflect the status of those libraries.

- A SYSMOD entry is created in the target zone for each SYSMOD that has been applied. Element entries (such as MOD and LMOD) are also created in the target zone for those elements that have been installed in the target libraries.
- SYSMOD entries in the global zone are updated to reflect that the SYSMOD has been applied to the target zone.
- BACKUP entries are created in the SMPSCDS data set so the SYSMOD can later be restored, if necessary.

3.8.2.1 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

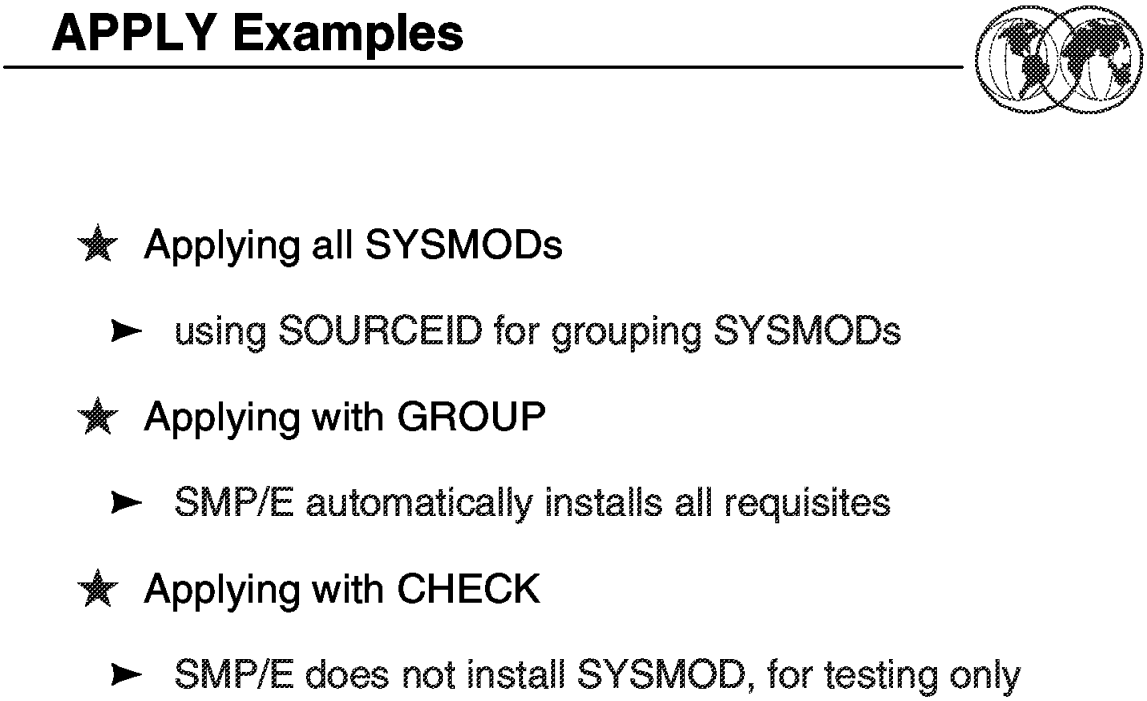


Figure 203. APPLY examples

3.8.2.2 APPLY examples

The following examples will help you to use the APPLY command.

3.8.2.3 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 PUT9901 and PUT9902. 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 use the sample JCL shown in Figure 204 on page 294 to perform this task.

```
//APPLY    JOB 'accounting info',MSGLEVEL=(1,1)
//APPLY    EXEC SMPPROC
//SMPTLIB  DD  UNIT=3380,VOL=SER=TLIB01
//SMPCNTL  DD  *
  SET      BDY(MVSTST1)            /* Process MVSTST1 tgt zone.  */.
  APPLY    SOURCEID(PUT9901,       /* Process these service      */
                    PUT9902)       /* levels                     */
           GROUP                   /* and any requisites.        */.
 /*
```

Figure 204. Sample JCL to apply all SYSMODs from a given source

3.8.2.4 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 commands shown in Figure 205.

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

Figure 205. Applying with the GROUP operand

3.8.2.5 Applying with the CHECK operand

In the next example, SMP/E was directed to automatically include SYSMODs needed for the selected function and service. At times, you may want to review which SYSMODs are included before you actually install them. This can be done by using the CHECK operand of APPLY, as shown in Figure 206.

```
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.             */.
```

Figure 206. Applying with the CHECK operand

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.

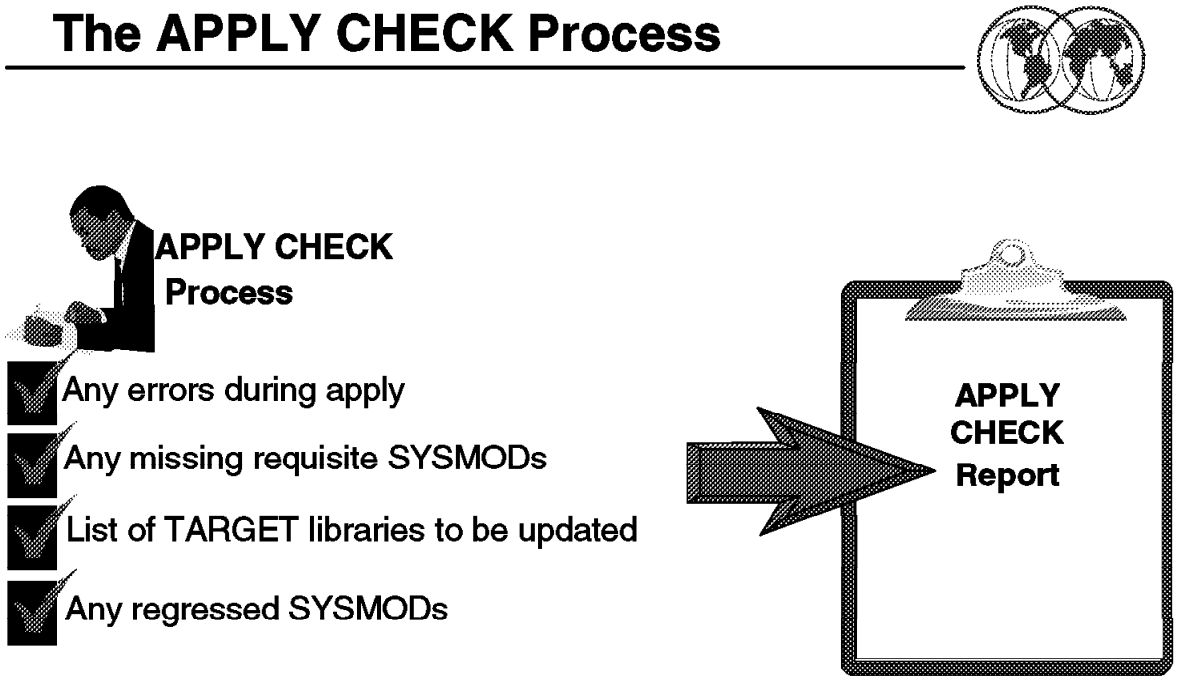


Figure 207. The APPLY CHECK process

3.8.3 The APPLY CHECK Process

The purpose of this step is to determine:

- Whether any errors will occur while the new function is being applied (except for errors that occur as a direct result of an update, such as a target library running out of space). This includes missing DDDEF entries.
- Whether any requisite SYSMODs are missing.
- List of target libraries that will be updated during the actual apply. This will allow you to back up those libraries before you apply the new elements.
- The SYSMODs, if any, that will be regressed.

3.8.3.1 Researching the APPLY CHECK reports

As a result of running the APPLY CHECK job, SMP/E produces various messages and reports that you should now use to do further research. Here are some of the errors that may have been detected:

- Some DD statements may be missing. Check the program directory or *OS/390 SMP/E Reference*, SC28-1806, to determine why they are required and how they should be specified.
- Some APAR fixes or USERMODs may be regressed. If so, you must determine why. For APAR fixes, you have to get the version of the APAR fix applicable to the new product. For USERMODs, you have to rework the modification to make it applicable to the new function, or eliminate the modification if the product being installed provides the same function. When doing the actual

APPLY operation, you may need to specify the BYPASS operand to inform SMP/E that you have resolved these problems.

- Some prerequisite or requisite PTFs may be missing. If so, you should determine whether they can be obtained. Some may already be on an ESO tape you have in-house but have not received; others may not have been shipped, in which case you have to get an early copy of them by contacting the IBM Support Center. Although you can also avoid these conditions by using the BYPASS operand, you are advised not to do this because the regressions have not been resolved.
- Some elements may not have been selected for installation. For each such element, if the current functional owner (that is, FMID) is an IBM product, there may not be a problem; this condition is common and occurs because there are multiple functions with common elements. Check the program directory or installation guide for the product you are installing to determine whether this condition is normal or if it indicates a problem.

  If the FMID is not one for an IBM product, further research is necessary. Contact the current owner of the element to determine how that product is related to the one you are installing.
- Some of the PTFs may not have been selected for installation because of exception SYSMOD conditions identified by the ++HOLD MCSs. When installing a new function, you may want to research these PTFs further. You can use the reason ID and the comments specified in the ++HOLD MCS to determine which of the following actions is most appropriate:
  - Bypass the condition using the BYPASS(HOLDERR) operand.
  - Do not install the PTF.
  - Obtain a fix for the APAR.

3.8.3.2 Getting additional SYSMODs

After doing the research step, you may decide that additional SYSMODs are needed. If so:

1. Obtain the additional SYSMODs by using CBPDO, ESO, CSSF Information/Access, SoftwareXcel Extended, or the IBM Support Center.
2. Receive the additional SYSMODs, using the same source ID value as used when processing the CUM tape.
3. Rerun the APPLY CHECK job.

Repeat this process until no errors are reported.

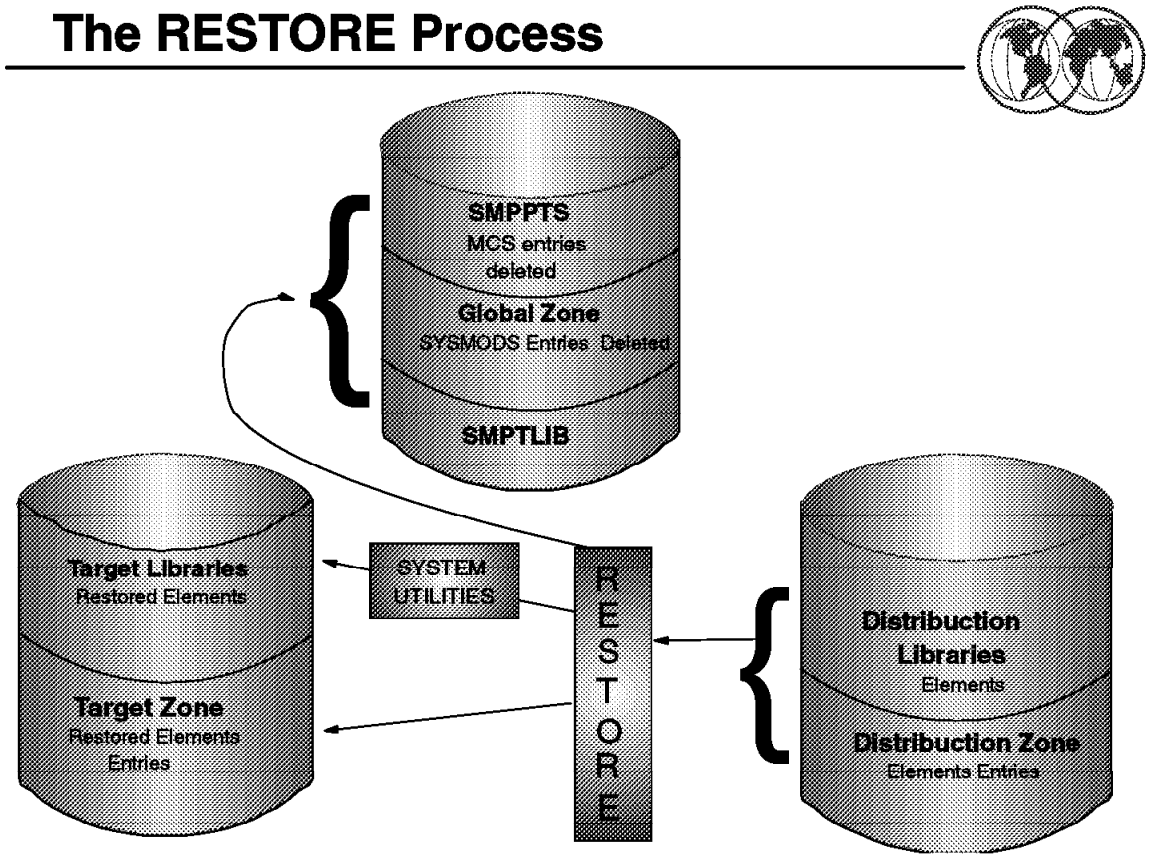


Figure 208. The RESTORE process

3.9 The RESTORE process

If you discover that a particular SYSMOD is causing a problem in your target libraries, you can remove it and replace the elements affected by it with the previous level of those elements, which is obtained from the backup (or distribution) libraries.

The RESTORE command processing replaces the elements on the target libraries with the last accepted version of the element that exists on the related distribution library (DLIB).

You can use the RESTORE command to remove SYSMODs from the target libraries and restore them to a previous level. The RESTORE command reverses APPLY processing, but has no effect on ACCEPT processing.

3.9.1 Removing SYSMODs

SMP/E ensures the eligibility of the selected SYSMODs and checks whether other SYSMODs are affected before continuing with RESTORE processing. Because of the various relationships and dependencies among the many SYSMODs, this checking is very important to the integrity of your system. In fact, to ensure that the requisites for a SYSMOD being restored are processed appropriately, SMP/E may require the whole chain of prerequisites to be restored.

3.9.2 Selecting elements

During RESTORE processing, SMP/E uses the information provided in the selected SYSMODs to determine which elements in the target zone should be replaced by elements in the related distribution libraries. The selection of elements is monitored by SMP/E to make sure that the correct functional level of each element is selected.

3.9.3 Replacing the elements in the target libraries

When SMP/E is satisfied that the proper SYSMODs have been selected, it uses information from the target zone to determine which distribution zone describes the elements necessary to replace the SYSMOD's elements in the target libraries. The RESTORE command directs SMP/E to call system utilities that replace the elements in the target libraries with the previous level of the elements from the related distribution libraries.

3.9.4 How SMP/E keeps track of RESTORE processing

SMP/E updates the information about the SYSMODs that have been restored. Remember, the target zone reflects the contents of the target libraries. Therefore, after the utility work is complete, and the target libraries have been updated, the target zone is updated to accurately reflect the status of those libraries.

- All information in the target zone pertaining to the restored SYSMOD is removed. The element entries in the target zone are restored to reflect the distribution zone level of the elements.
- The global zone SYSMOD entries and MCS statements, which are stored in the SMPPTS data set, are deleted for those SYSMODs that have been restored. Any SMPTLIB data sets created during RECEIVE processing are also deleted for the restored SYSMOD. SMP/E automatically performs this global zone clean-up, unless you specify otherwise.

3.9.5 The RESTORE command

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.)

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.

Some of these conditions are described next. For more detail, see the topic "Usage Notes" in *OS/390 SMP/E Commands*, SC28-1805.

- 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.
- 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.

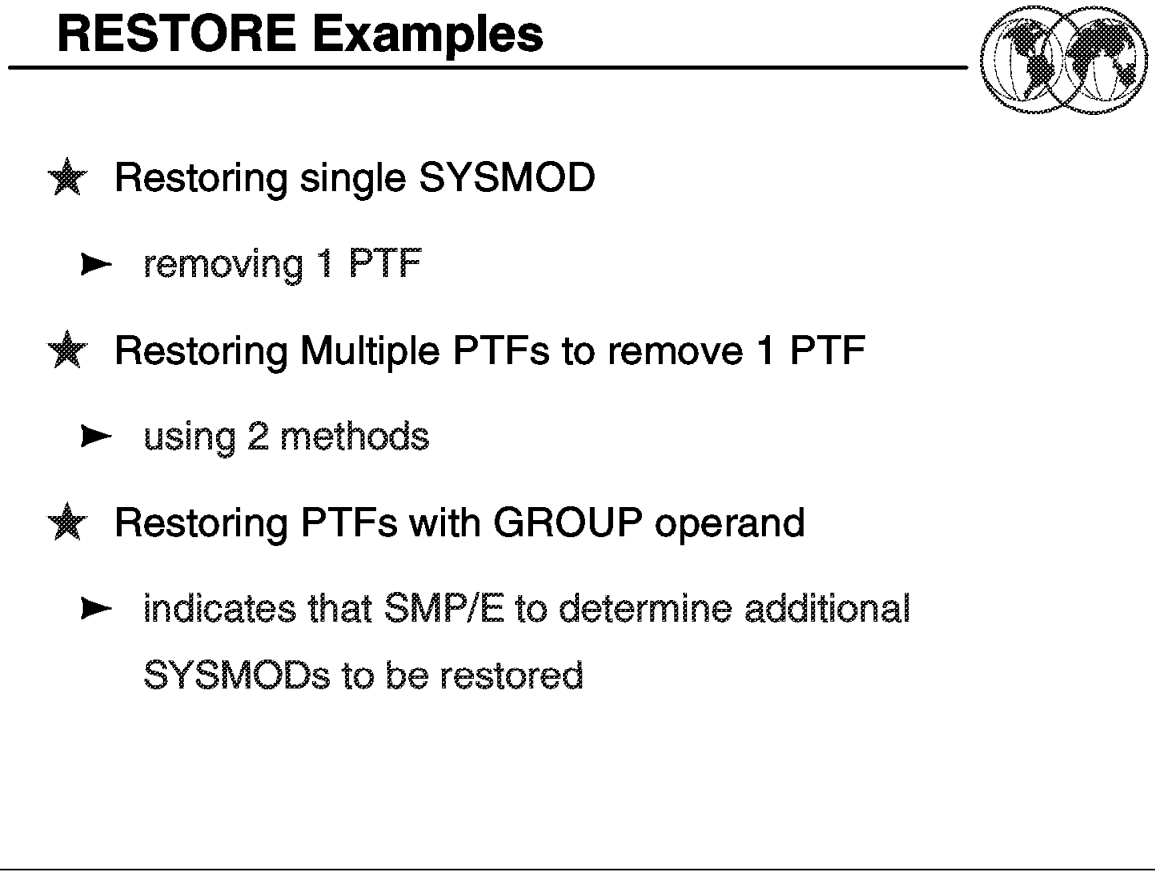


Figure 209. RESTORE examples

3.9.6 Restore examples

The following examples can help you to use the RESTORE command.

3.9.6.1 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. You can use the RESTORE command shown in Figure 210 to do this task.

```
SET      BDY(TGT1)          /* Set to target zone.      */.
RESTORE  S(UZ00001)         /* Restore 1 PTF.           */.
```

Figure 210. Removing a single 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 completes as shown in Figure 211 on page 300.

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
REJECT   S(UZ00001)         /* Reject 1 PTF.            */.
```

Figure 211. Cleaning up the SMP records after the reject

3.9.6.2 Restoring multiple PTFs to remove one PTF

Assume you have applied the PTFs named 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 (we can say that the RMID of the module is UZ00003), you want to restore that PTF from the system.

You have two choices:

1. Restore PTF UZ00001, UZ00002, UZ00003, and then reapply UZ00001 and UZ00002 as shown in Figure 212.

```
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.       */.
```

Figure 212. Restoring and reapplying PTFs

2. Accept PTFs UZ00001 and UZ00002, if you are sure that they have no errors, then restore UZ00003 as shown in Figure 213.

```
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.*/.
```

Figure 213. Accepting some PTFs and then restoring another

The end result in both cases is that module XYMOD01 from PTF UZ00002 is in the target libraries.

3.9.6.3 Restoring PTFs using the GROUP operand

In 3.9.6.2, “Restoring multiple PTFs to remove one PTF,” 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 commands shown in Figure 214 on page 301 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. */.
```

Figure 214. Restoring PTFs using the Group operand

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.

3.9.6.4 The reporting output

When RESTORE processing is complete, these reports will help you analyze the results:

- The *SYSMOD Status report* provides you with a summary of the processing that took place for each eligible SYSMOD, based on the operands you specified on the RESTORE command. It shows you which SYSMODs were restored, which were not restored, and why they were not restored.
- The *Element Summary report* provides you with a summary of the processing that took place for each eligible SYSMOD, based on the operands you specified on the RESTORE command. It shows you which SYSMODs were restored, which were not restored, and why they were not restored.
- The *Causer SYSMOD Summary report* report provides you with a list of SYSMODs that caused other SYSMODs to fail, and describes the errors that must be fixed to successfully process the SYSMODs. This report can reduce the amount of work involved in figuring out which errors caused SYSMODs to fail.
- The *File Allocation report* provides you with a list of the data sets used for RESTORE processing and supplies information about these data sets.

Additional reports may be produced depending on the work being done and the content of the SYSMODs. For more information about all the reports produced by the RESTORE command (and samples of actual reports), see *OS/390 SMP/E Commands*, SC28-1805.

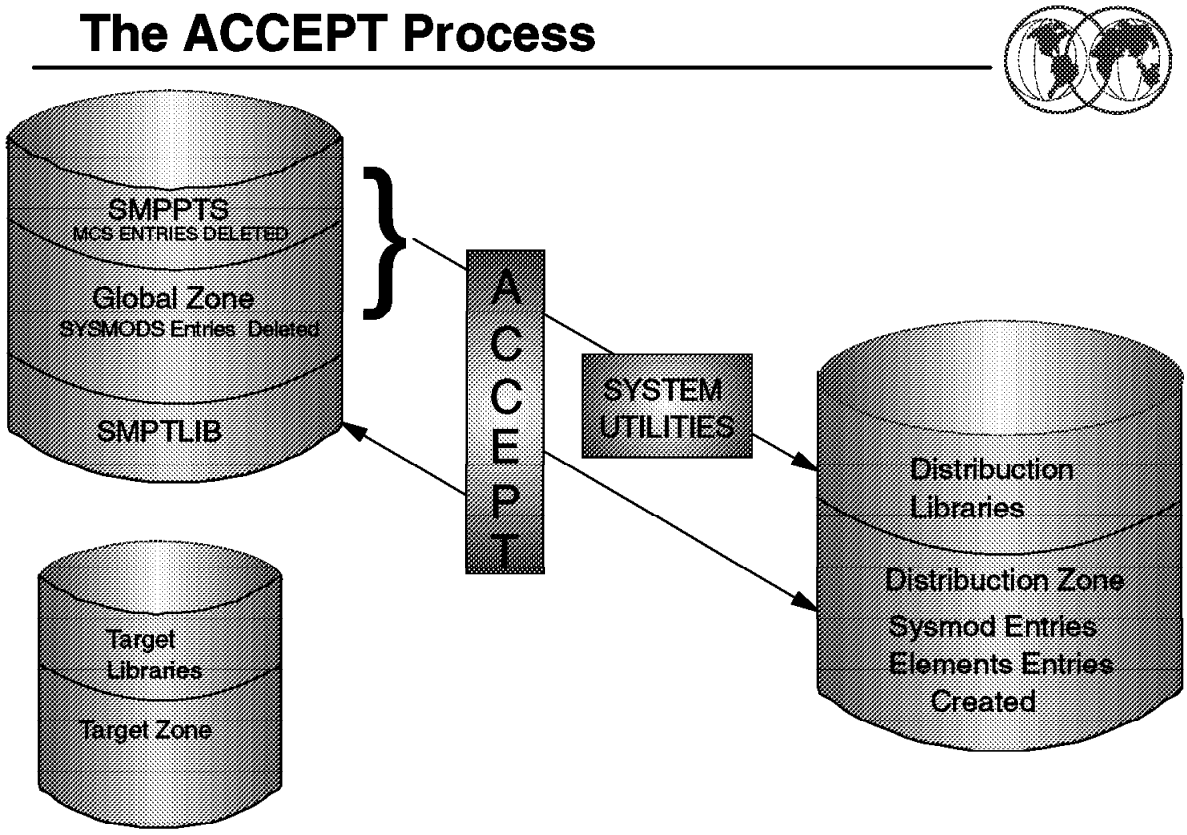


Figure 215. The ACCEPT process

3.10 The ACCEPT process

You can use the ACCEPT process to install a SYSMOD in a backup (or distribution library). The ACCEPT process is very similar to the APPLY process with one important exception—it is irreversible.

After you are satisfied that an applied SYSMOD has performed reliably in your target system, you can install it in your backup system (distribution) libraries.

When you accept a PTF, you must know that if in the future you need to restore a PTF that changes the same module of the accepted PTF, this module will be copied from the DLIB to your target library and it will retrograde to the level of the last PTF accepted. If you have never accepted a PTF, the module will retrograde to the base level, or at least, to the level that was installed.

3.10.1 Selecting SYSMODs

You can specify operands on the ACCEPT command that tell SMP/E which of the received SYSMODs are to be selected for installation in the distribution libraries. SMP/E ensures that all other required SYSMODs have been installed or are being installed concurrently and in the proper sequence.

3.10.2 Selecting elements

During ACCEPT processing, SMP/E uses the information provided in the selected SYSMODs to determine which elements should be installed in the distribution libraries. The selection of elements is monitored by SMP/E to make sure that the correct functional level of each element is selected.

3.10.3 Updating the distribution libraries

After the proper SYSMODs have been selected and the proper functional and service level of each element has been checked, SMP/E calls the system utilities (in the same manner as APPLY and RESTORE) to place the elements into the distribution libraries described in the distribution zone. The source of the elements is the SMPTLIB data sets, the SMPPTS data set, or the indirect libraries, depending on how the SYSMOD was packaged.

3.10.4 How SMP/E keeps track of ACCEPT processing

SMP/E updates the information about the SYSMODs that have been accepted. Remember, the distribution zone reflects the contents of the distribution libraries. Therefore, after the utility work is complete, and the distribution libraries have been updated, the distribution zone is updated to accurately reflect the status of those libraries.

- A SYSMOD entry is created in the *distribution zone* for each SYSMOD that has been accepted. Element entries (such as MOD and LMOD) are also created in the distribution zone for the elements that have been installed in the distribution libraries.
- *Global* zone SYSMOD entries and MCS statements in the SMPPTS data set are deleted for those SYSMODs that have been accepted into the distribution zone. Any SMPTLIB data sets created during RECEIVE processing are also deleted. If you do not want SMP/E to do this global zone clean-up, you have the option to indicate this to SMP/E, and the information is saved.

3.10.5 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

3.10.6 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.

We recommend that you always use the ACCEPT CHECK prior to accepting the SYSMODs.

3.10.7 The reporting output

When ACCEPT processing is complete, these reports will help you analyze the results:

- The *SYSMOD Status report* provides you with a summary of the processing that took place for each eligible SYSMOD, based on the operands you specified on the ACCEPT command. It shows you which SYSMODs were accepted, which were not accepted, and why they were not accepted.
- The *Element Summary report* provides you with a detailed look at each element affected by ACCEPT processing. It tells you in which libraries the elements were installed.
- The *Causer SYSMOD Summary report* provides you with a list of SYSMODs that caused other SYSMODs to fail, and describes the errors that must be fixed to successfully process the SYSMODs. This report can reduce the amount of work involved in figuring out which errors caused SYSMODs to fail.
- The *File Allocation report* provides you with a list of the data sets used for ACCEPT processing and supplies information about these data sets.

Additional reports may be produced depending on the work being done and the content of the SYSMODs. For more information about all the reports produced by the ACCEPT command (and samples of actual reports), see *OS/390 SMP/E Commands*, SC28-1805.

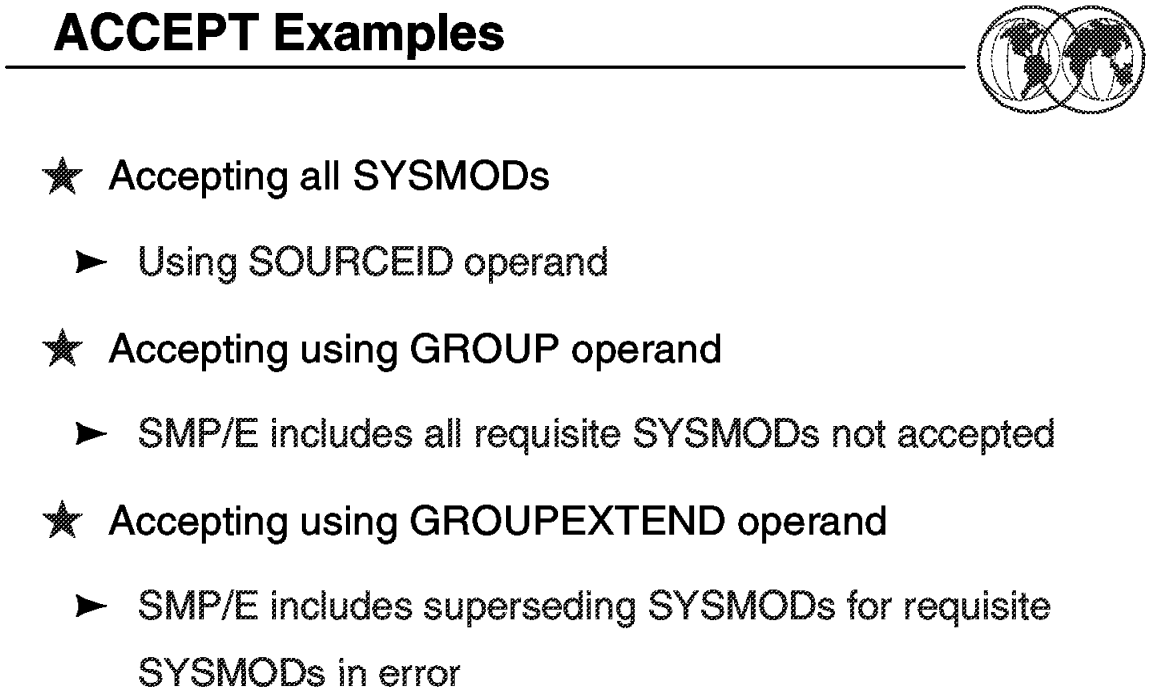


Figure 216. ACCEPT examples

3.10.8 ACCEPT examples

The following examples will help you with the ACCEPT command.

3.10.8.1 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 PUT9901 and PUT9902. 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 commands shown in Figure 217.

```
SET      BDY(MVSDLB1)       /* Process MVSDLB1 DLIB zone. */.
ACCEPT   SOURCEID(PUT9901,  /* Process these service      */
                 PUT9902)   /* levels                     */
         GROUP              /* and any requisites.        */.
```

Figure 217. Accepting all SYSMODs from a given source

3.10.8.2 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, HPT1234, with all its service and any requisite SYSMODs. You can do this with the commands shown in Figure 218 on page 306.

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

Figure 218. Accepting SYSMODs with ACCEPT

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.

3.10.8.3 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 as shown in Figure 219.

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

Figure 219. Using the GROUPEXTEND operand

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.

Note: You must be careful in using the ACCEPT command. When you accept a SYSMOD, you cannot restore it by using SMP/E.

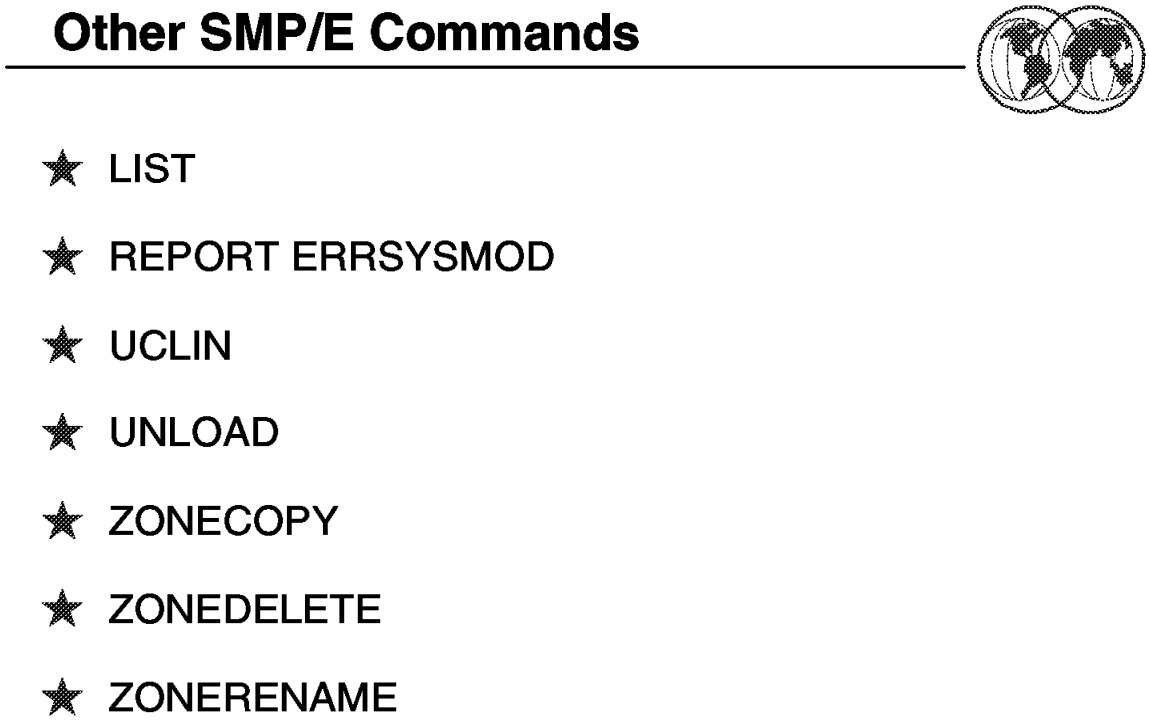


Figure 220. Other SMP/E commands

3.11 Other useful SMP/E commands

In the course of managing your system, 90 percent of the time you will only use the RECEIVE, APPLY, and ACCEPT commands. However, there are times when you need to use other SMP/E commands to perform a specific task, such retrieving information from SMP/E data sets, or cloning a new target zone from the existing target zone.

Some of the more useful SMP/E commands are:

- LIST command

  You can use the LIST command to retrieve information stored in SMP/E data sets.

- REPORT ERRSYSMOD command

  This command helps you to identify exception SYSMODs and also any resolving SYSMODs for the held SYSMODs.

- UCLIN command

  With the UCLIN command, you can add, delete, or replace entries in the following SMP/E data sets:

  - SMPCSI
  - SMPMTS
  - SMPSCDS
  - SMPSTS

The UCLIN command only updates entries in SMP/E data sets, hence it does nothing to any elements or load modules in any product libraries.

Note: Be sure you understand the relationships between the various entries before you make any UCLIN changes, as this helps to ensure that any UCLIN changes made are complete and consistent with one another. Remember, SMP/E does not check how the UCLIN changes might affect the other entires.

- UNLOAD command

  Sometimes you want to make changes to a set of DDDEF entries in the target zone. Instead of using the SMP/E dialog and making changes to each DDDEF entry one by one, you can use the UNLOAD command which causes SMP/E to unload all the DDDEF entries from the target zone. The output produced is in the form of UCLIN statements, to which you can make the necessary updates for the DDDEF entries.

- ZONECOPY command

  The ZONECOPY command can be used to copy an entire target or distribution zone from an existing CSI data set to another CSI data set. SMP/E copies the data from the input zone to the other CSI and renames the receiving zone. This is useful when you need to set up a new target or distribution zone based on the existing zones.

- ZONEDELETE command

  There are times when you want to delete the information about an old level of the product after installing a new level of product in its own target and distribution zone.

- ZONERENAME command

  Sometimes the current name of a zone does not conform to the naming standard established in your environment and thus you want to assign a new name to the zones. You can do that by using the ZONERENAME command which allows you to change the name of an existing target or distribution zone.

In this section, we will only introduce the usage of the LIST and REPORT ERRSYSMODS commands. For more information on the rest of the commands, see *OS/390 SMP/E Commands*, SC28-1805.

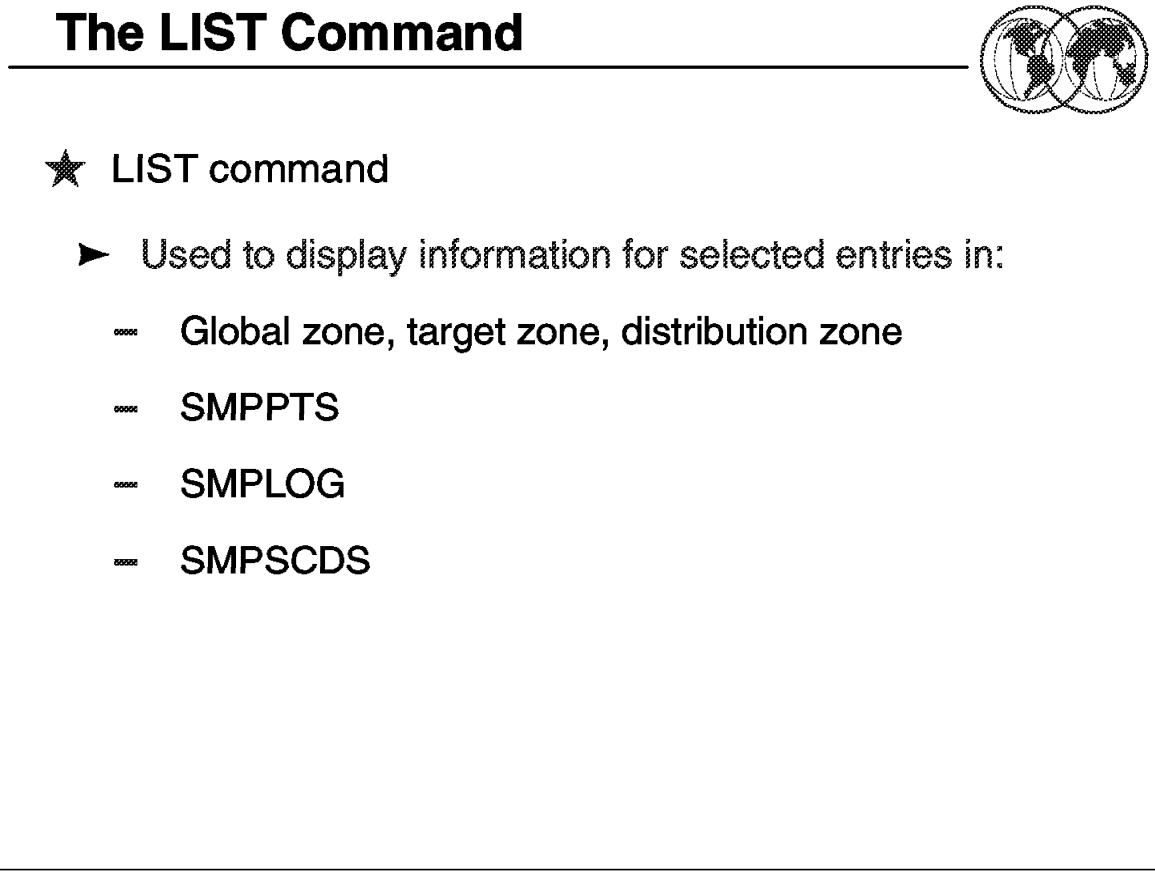


Figure 221. The LIST command

3.11.1 Using the LIST command

The SMP/E data sets (the global zone, target zones, distribution zones, SMPPTS, SMPLOG, and SMPSCDS) contain a great deal of information that you may find useful when installing a new function, preparing a user modification, or debugging a problem. You can use the LIST command to display that information.

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 sets to be listed.

Note: Make sure that the data you request to be listed is valid for the specified zone type.

3.11.1.1 Listing entries in a particular zone

There are times when you need to check all the DDDEF entries defined in the target zone. You can use the sample JCL shown in Figure 222 on page 310 to perform this task.

```
//LIST1   JOB (),'MVSSP',NOTIFY=&SYSUID,CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=X
//SMPE     EXEC PGM=GIMSMP
//SMPCSI   DD DSN=SMPE.GLOBAL.CSI,DISP=SHR
//SMPRPT   DD SYSOUT=*
//SMPOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SMPCNTL  DD *
  SET BDY (MVST100).    /* Set to target zone     */
  LIST DDDEF       .    /* List all DDDEF entries */
```

Figure 222. Listing entries in a particular zone

Note: If you want to check all the DDDEF entries defined in the global zone and all the zones defined to that global zone, you can use the LIST command shown in Figure 223 (the global and any zone defined to it can be used in the SET BOUNDARY command).

```
SET BDY(MVST100) .
LIST DDDEF ALLZONES .
```

Figure 223. Listing all DDDEF entries in the global zone and all defined zones

Suppose you want find out all the PTFs that have been applied to the target zone for the function SYSMOD HTCP340. You can use the command shown in Figure 224 to accomplish the task:

```
SET BDY(MVST100) .
LIST SYSMOD PTFS .
```

Figure 224. Listing all PTFs for a specific SYSMOD

Note: If you just use the SYSMOD operand, SMP/E lists all SYSMOD entries in that zone specified in the SET BOUNDARY command. You can limit which SYSMOD entries are listed by coding one or more SYSMOD qualifiers, such as APARS, PTFS, FUNCTION, and so forth.

3.11.1.2 Reports

At the end of the LIST processing, two reports are generated:

- The File Allocation report
- The LIST Summary report

For more information about these reports, see *OS/390 SMP/E Commands*, SC28-1805.

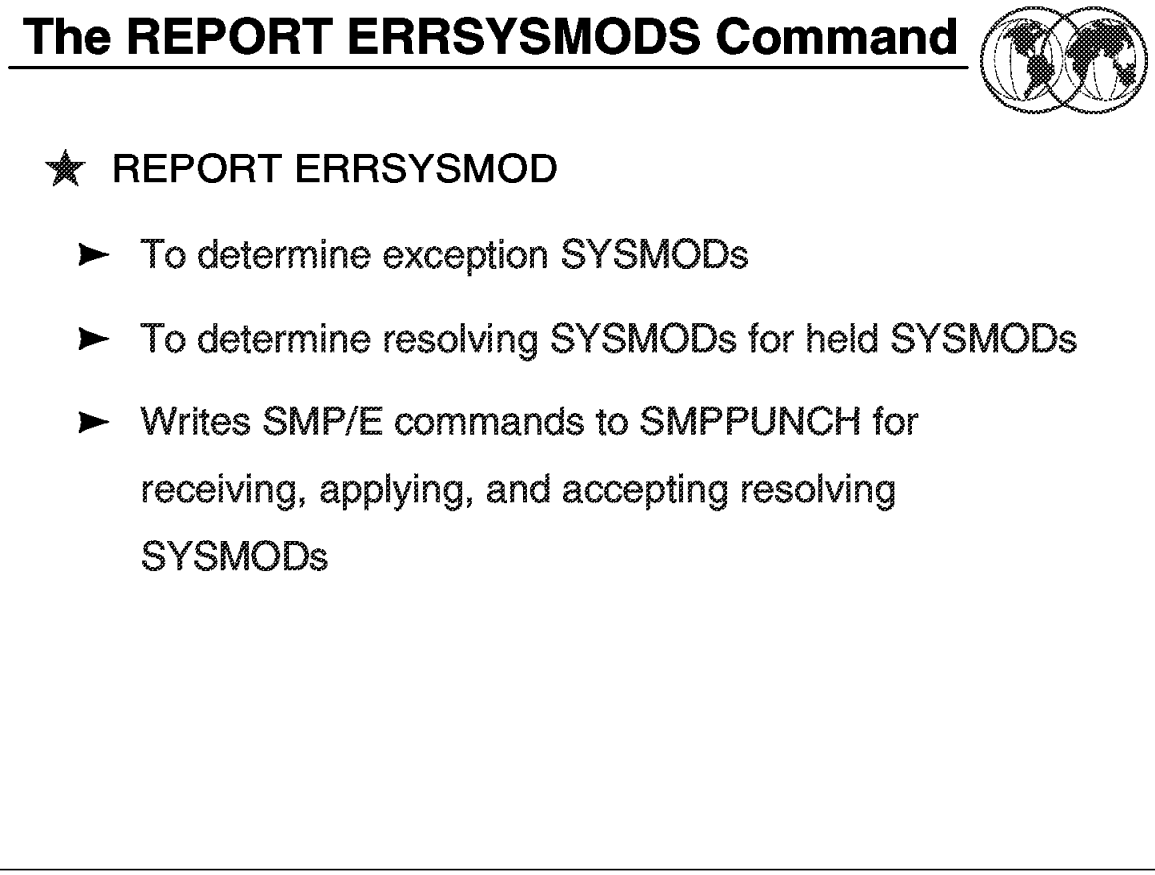


Figure 225. The REPORT ERRSYSMODS command

3.11.2 Using the REPORT ERRSYSMOD command

This command helps you to determine whether any SYSMODs you have already installed are now exception SYSMODs. It also helps you to determine whether any resolving SYSMODs are available for held SYSMODs.

Using REPORT ERRSYSMODS for target and distribution zones, the command lists installed SYSMODs for which ++HOLD statements were subsequently received and whose error reason IDs have not yet been resolved.

Using REPORT ERRSYSMODS for a global zone, the command lists received SYSMODs for which ++HOLD statements with error reason IDs have been received.

Note: When using the REPORT ERRSYSMODS command, the SET BOUNDARY command must specify GLOBAL.

3.11.2.1 Examples of the REPORT ERRSYSMODS command

The following are some examples showing how you can use the REPORT ERRSYSMODS command to get the information that you want:

- Assume you have received the latest HOLDDATA, and you want to know whether it affects any of the SYSMODs that have already been applied to both the target and distribution zone. You can use the command shown in Figure 226 on page 312 to accomplish the task:

```
SET    BDY(GLOBAL) 1 .
REPORT ERRSYSMODS
       ZONES(MVST100,MVSD100) 2
       BEGINDATE(06 01 98) 3
       ENDDATE(04 01 99) .
```

Figure 226. Checking if HOLDDATA affects any already applied SYSMODs

1 Remember you have to specify GLOBAL in the SET BDY command when you use the REPORT ERRSYSMODS command.

2 Specify the name of target and distribution zone to report on.

3 The BEGINDATE and ENDDATE indicates the begin and end date of the ++HOLD statement received by SMP/E for the REPORT ERRSYSMODS processing. The format of the date is *mm dd yy*.

- There are situations where you are only interested in the effect of the HOLDDATA on a certain function SYSMOD, for example HBB6607. You can use the FORFMID operand to achieve the objective as is shown in Figure 227.

```
SET    BDY(GLOBAL)
REPORT ERRSYSMODS
       ZONES(MVST100)
       FORFMID(HBB607) 1
       BEGINDATE(06 01 98)
       ENDDATE(04 01 99) .
```

Figure 227. Checking the effect of HOLDDATA on a specific SYSMOD

1 Specifying the FORFMID operand will limit the list of SYSMODs to those that have the ++HOLD statement with the FMID specified.

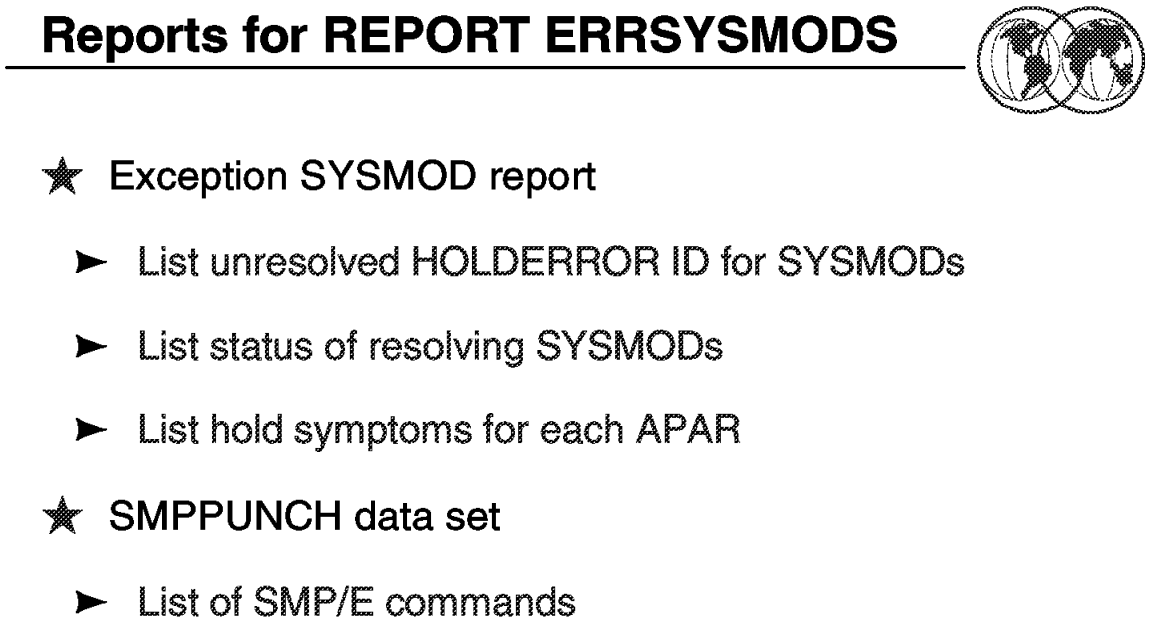


Figure 228. Reports for REPORT ERRSYSMODS

3.11.2.2 Reports for REPORT ERRSYSMODS

At the end of the REPORT ERRSYSMODS processing, information about the held SYSMODs and any resolving SYSMODs is written to the Exception SYSMOD report. At the same time, commands needed to install the resolving SYSMODs are provided in the SMPPUNCH data set.

- 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 (APAR numbers) that have made them exception SYSMODs, resolving SYSMODs that have not yet been installed, and the hold class and hold symptoms for each APAR.

  The exception SYSMOD reports produced by a given REPORT ERRSYSMODS command are arranged alphanumerically by zone name. Each report begins on a new page. The information gathered for each zone is sorted in ascending order, first by FMID, then by SYSMOD name, then APAR number, and finally by resolving SYSMOD.

  Figure 229 on page 314 shows an example of exception SYSMOD report:

```
*************************** TOP OF DATA ********************************
PAGE 0001  - NOW SET TO GLOBAL ZONE         DATE 04/21/99  TIME 10:16:50

EXCEPTION SYSMOD REPORT FOR ZONE MVST100      DATE: 01/01/98 - 03/01/99

     HOLD    SYSMOD   APAR     ---RESOLVING SYSMOD---- HOLD   HOLD
     FMID    NAME     NUMBER   NAME    STATUS RECEIVED CLASS  SYMPTOMS
     ---------------------------------------------------------------------

 1  HMJ4102  UW31189  AN80203  UW32213 GOOD   YES      PE     IPL
 2  HQA5140  UW42146  AN90025  UW43610 HELD   NO       PE

PAGE 0002  - NOW SET TO GLOBAL ZONE          DATE 04/21/99  TIME 10:16:5

EXCEPTION SYSMOD REPORT SUMMARY             DATE: 01/01/98 - 03/01/99

 ZONE    FMID        TOTAL APARS 3  TOTAL RESOLVING 4
                    AGAINST FMID      SYSMODS AGAINST FMID
-------------------------------------------------------------------------
MVST100  HMJ4102            1                  1
         HQA5140            1                  1
```

Figure 229. Sample Exception SYSMOD Report

1 UW31189 is an exception SYSMOD for function HMJ4102. The reason for being held is that it is a PE (PTF in error). However, there is a resolving SYSMOD UW32213 which has already been received and has no known problems.

2 UW42146 is a PE for function HQA5140, and the supposed resolving SYSMOD, UW43610, is being held for error described in APAR AN90025.

3 This indicates the total number of APARs that have error holds against the FMID.

4 This indicates the total number of resolving SYSMODs found against the FMID. The number includes all received APARs and all PTFs (both received and unreceived, held and unheld) against the FMID.

- The SMPPUNCH data set

  In order to help you to install resolving SYSMODs for exception SYSMODs, SMP/E writes the necessary commands to the SMPPUNCH data set:

  - SET BOUNDARY
  - RECEIVE (for unreceived resolving SYSMODs)
  - RESETRC (sets the return codes for the preceding commands to zero)
  - Either APPLY (for target zones) or ACCEPT (for distribution)

  However, there are cases where nothing is written to SMPPUNCH for a specified zone:

  - There are no exception SYSMODs for the specified zone.
  - There are no resolving SYSMODs for any of the exception SYSMODs, or all resolving SYSMODs identified are held.
  - The specified zone is the global zone.
  - NOPUNCH operand was specified on the REPORT ERRSYSMODS command.

  Figure 230 on page 315 shows an example of SMPPUNCH output.

```
************************** TOP OF DATA ******************************
SET BDY (GLOBAL ). /* REMOVE COMMENT IF DOING RECEIVE
RECEIVE  SELECT (
         UW50483
         UW50095
         UW49599
            .
            .
            .
         UW55773
         UW56914 )
          SYSMODS.
                   REMOVE COMMENT IF DOING RECEIVE */
RESETRC.
SET BDY (MPRDTZ ).
APPLY    SELECT (
 /* UW50483             RESOLVES AW29065 FOR HBB6605 FMID(HBB6605) */
    UW52639 /* PTF      RESOLVES AW29065 FOR HBB6605 FMID(HBB6605) */
       .
       .
       .
    UW53498 /* PTF      RESOLVES AW35984 FOR UW51679 FMID(HPRF220) */
                )
          GROUP.
******************************** BOTTOM OF DATA ************************
```

Figure 230. Sample SMPPUNCH output

For more information on the LIST, REPORT ERRSYSMODS and the rest of the SMP/E commands, see *OS/390 SMP/E Commands*, SC28-1805.

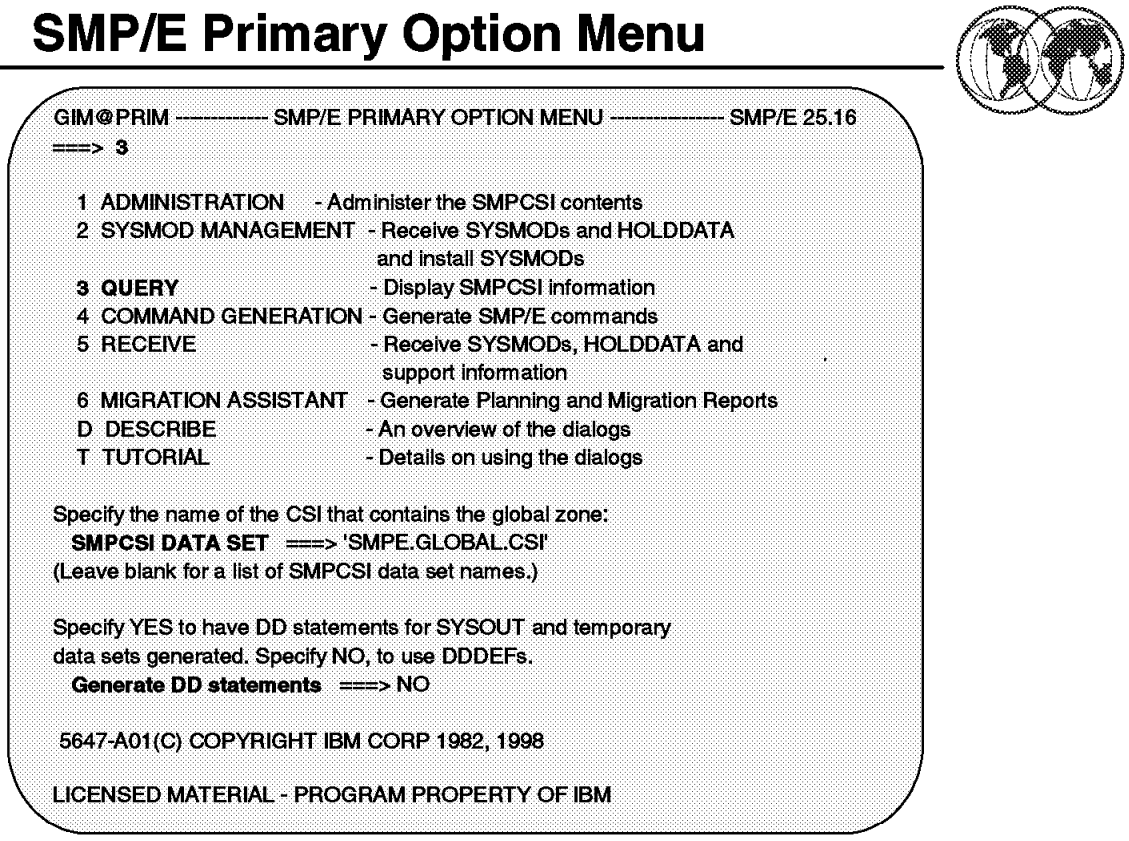


Figure 231. SMP/E Primary Option Menu

3.12 SMP/E dialogs

The SMP/E Primary Option Menu is the entry point to the SMP/E dialogs. It is on this panel that you select which path of the dialogs you will be using. The global zone which is to be used by the dialogs is specified in this screen.You *must* specify a global zone in this panel.

The SMP/E dialogs provide you with an online method of system management, software inventory, data base inquiries, and guidance. For example, with the Query dialogs, you can look up information in the CSI data set. The Query dialogs are one of the easiest and most direct methods you can use to obtain the content and status of any SYSMOD that has been processed by SMP/E. You can use the Query dialogs to display an entry in either a specific zone (CSI query) or in all zones (cross-zone query).

To get to the Query dialogs, you select SMP/E (option 1) on the initial SMP/E dialog panel (CIDPGV2). Then, on the main menu for SMP/E options (GIM@PRIM), select Query (option 3). This takes you to the initial Query panel. If you need assistance with using the Query dialogs (or any of the SMP/E dialogs), help panels are available.

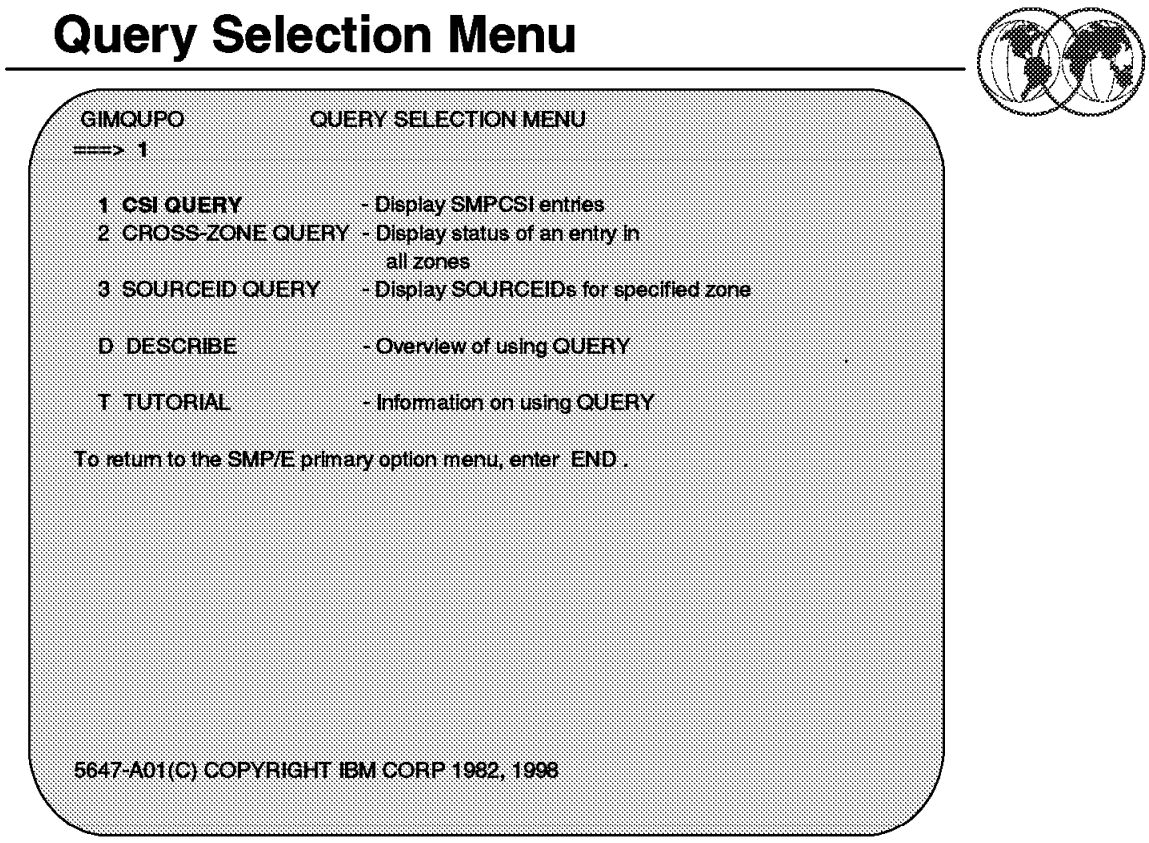


Figure 232. Query Selection Menu

3.12.1 Query Selection Menu

Let's assume you want to find out which SYSMODs have been applied to a particular target zone on your system. You can accomplish this task using the QUERY SELECTION MENU and selecting the CSI QUERY option (1).

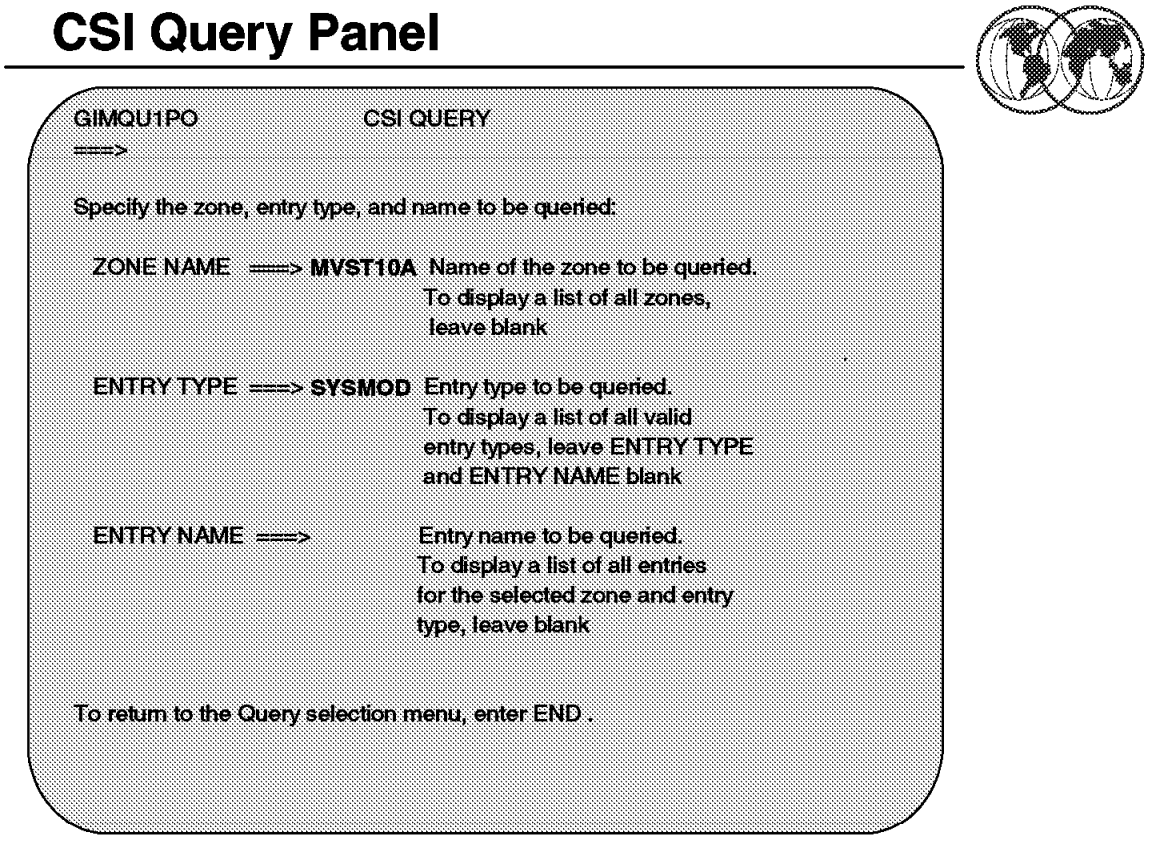


Figure 233. CSI Query Panel

3.12.2 CSI Query Panel

When the CSI QUERY panel is displayed, you can indicate that you want SMP/E to check target zone MVST10A for all SYSMOD entries.

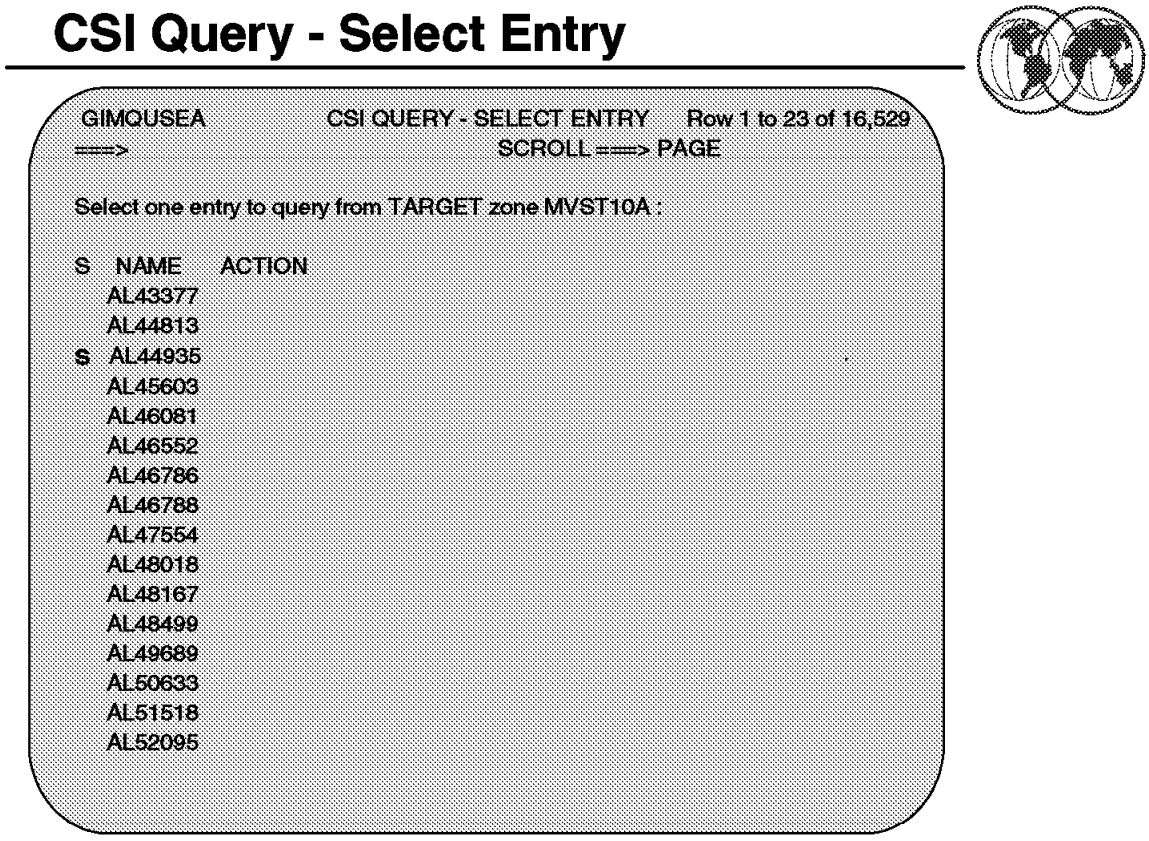


Figure 234. CSI Query - Select Entry Panel

3.12.3 CSI Query - Select Entry Panel

Because the ENTRY NAME was left blank on the CSI QUERY panel, SMP/E displays another panel (see the visual) that lists all the SYSMOD entries in target zone MVSTGT1.

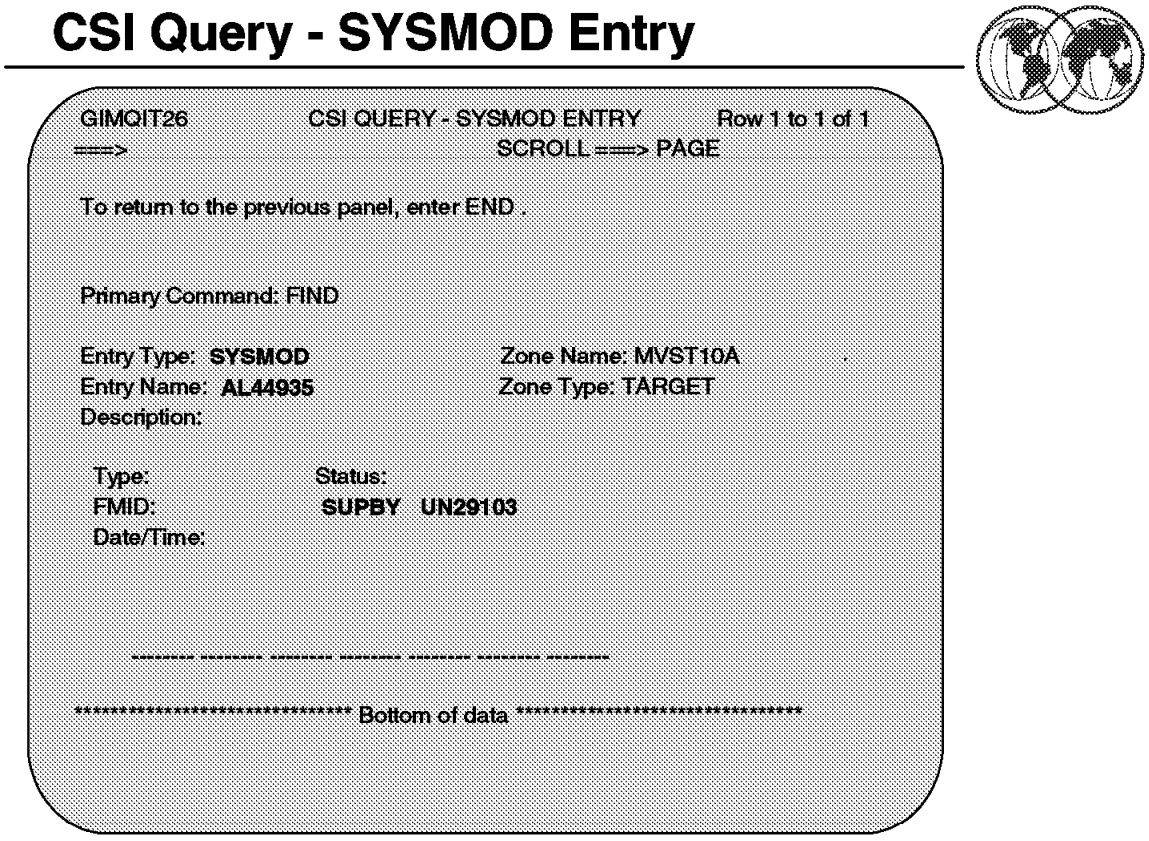


Figure 235. CSI Query - SYSMOD Entry Panel

3.12.4 CSI Query - SYSMOD Entry Panel

This panel shows all relevant information about the SYSMOD AL44935. As you can see, the PTF AL44935 is superseded by UN29103; this means that PTF UN29103 has incorporated the code of AL44935.

As you can see, the QUERY dialog panels provide a quick and easy way for you to obtain information about your system.

Building SMP/E Jobs by using Dialog

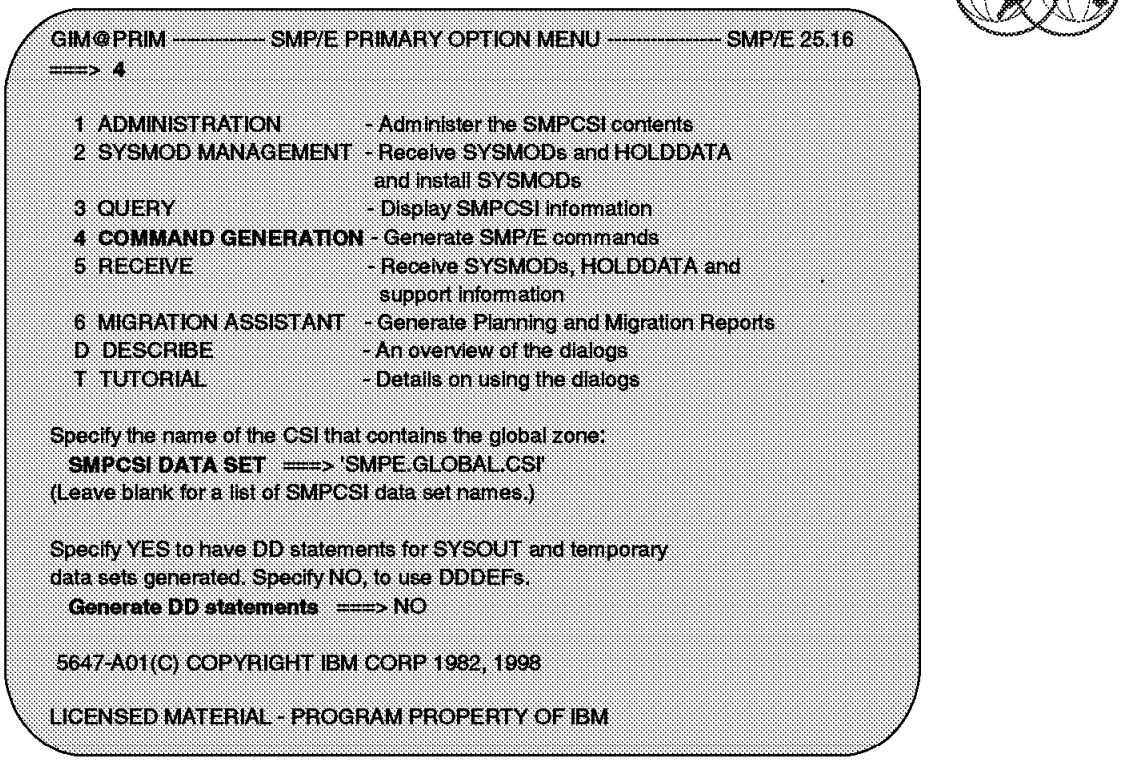


Figure 236. Building SMP/E jobs by using dialog

3.12.5 Building SMP/E jobs by using dialog

You can use the SMP/E dialog to generate jobs, for RECEIVE, APPLY, LIST, and so forth.

Suppose that you want to list all PTFs that are RECEIVED for an FMID. In the SMP/E Primary Option Menu, you select Option (4), COMMAND GENERATION. You must also enter the SMPCSI DATA SET field, or enter it on the next displayed panel. If you specify the name of the CSI, you receive the panel on the next visual.

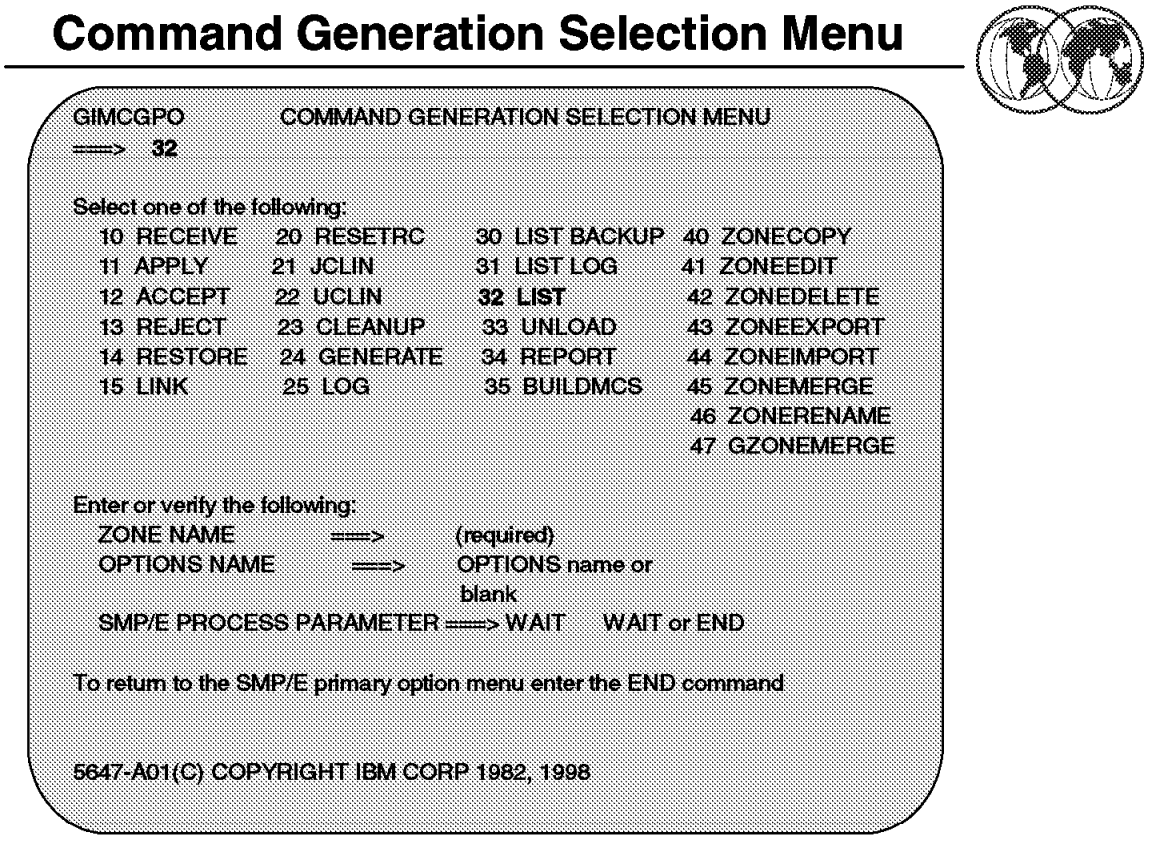


Figure 237. Command Generation Selection Menu

3.12.6 Command Generation Selection Menu

Using this panel, SMP/E lists the commands that you can select. For an example of the use of this panel, we will use OPTION (32) 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.

Command Generation Selection Zone

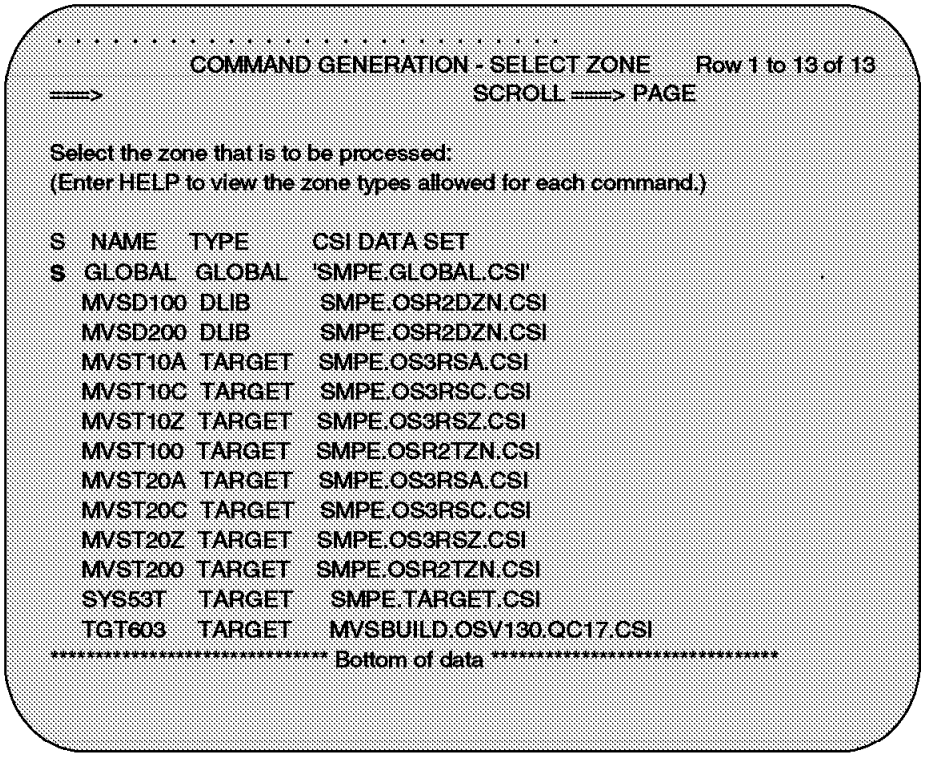


Figure 238. Command Generation Selection Zone

3.12.7 Command Generation Selection Zone

Select the CSI data set that you want to use; in our case, shown in the visual is the GLOBAL CSI.

Command Generation - List Command

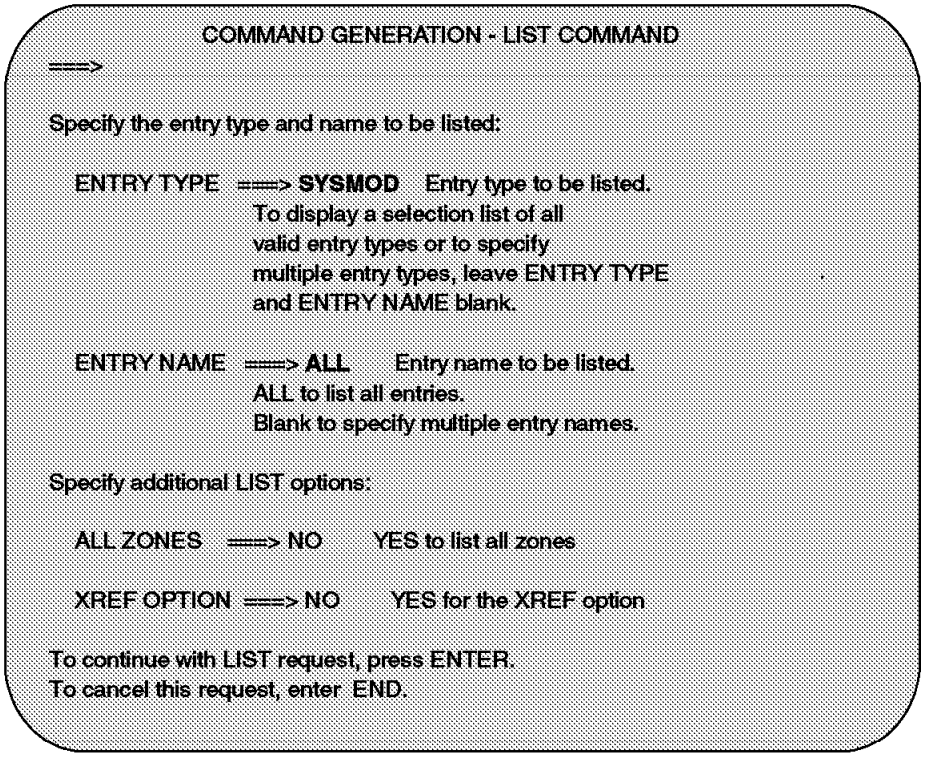


Figure 239. Command Generation - LIST Command

3.12.8 Command Generation - LIST Command

In the visual, We selected ENTRY TYPE=SYSMOD and ENTRY NAME=ALL.

Note: If you left ENTRY TYPE or ENTRY NAME blank, SMP/E will display a panel with all entry types and entry names listed and will ask you to make a selection.

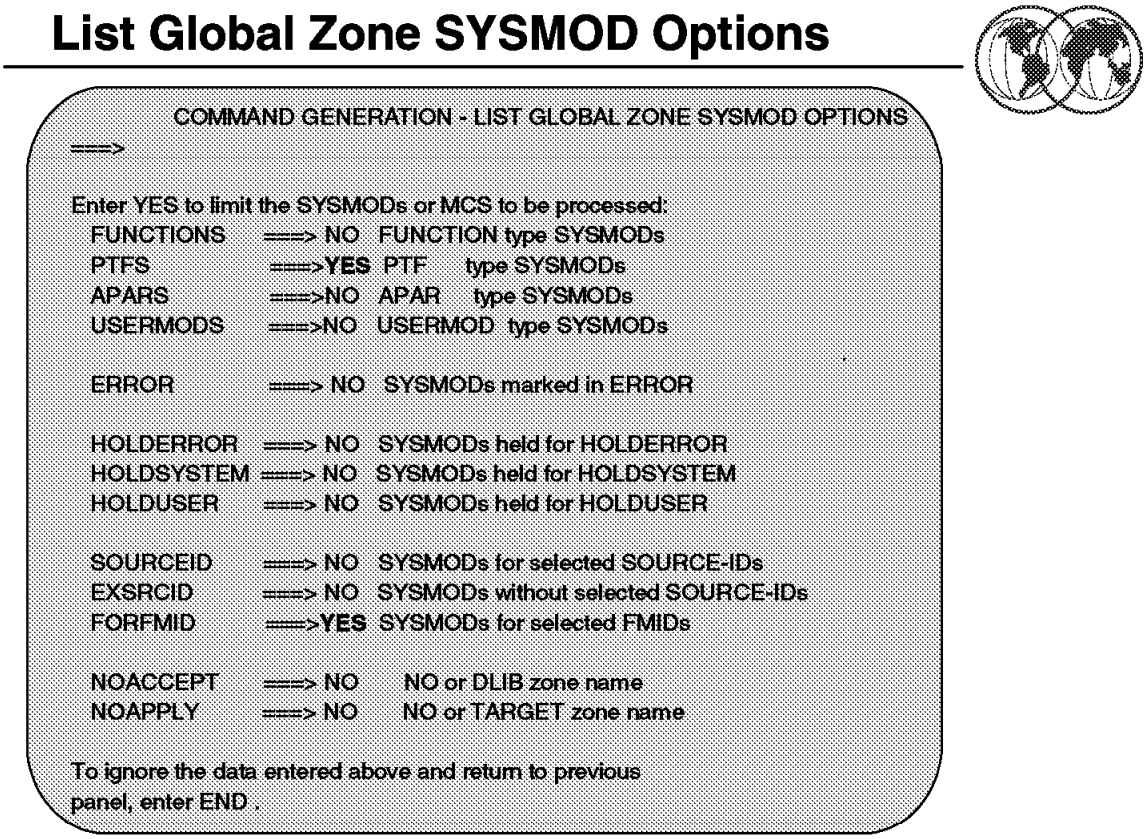


Figure 240. List Global zone SYSMOD options

3.12.9 List Global zone SYSMOD options

In the List Global Zone Option, we will select PTFs by entering YES, as shown in the visual (because we want to list all PTFs that are received), and FORFMID as YES (because we want to restrict the search for only two FMIDs).

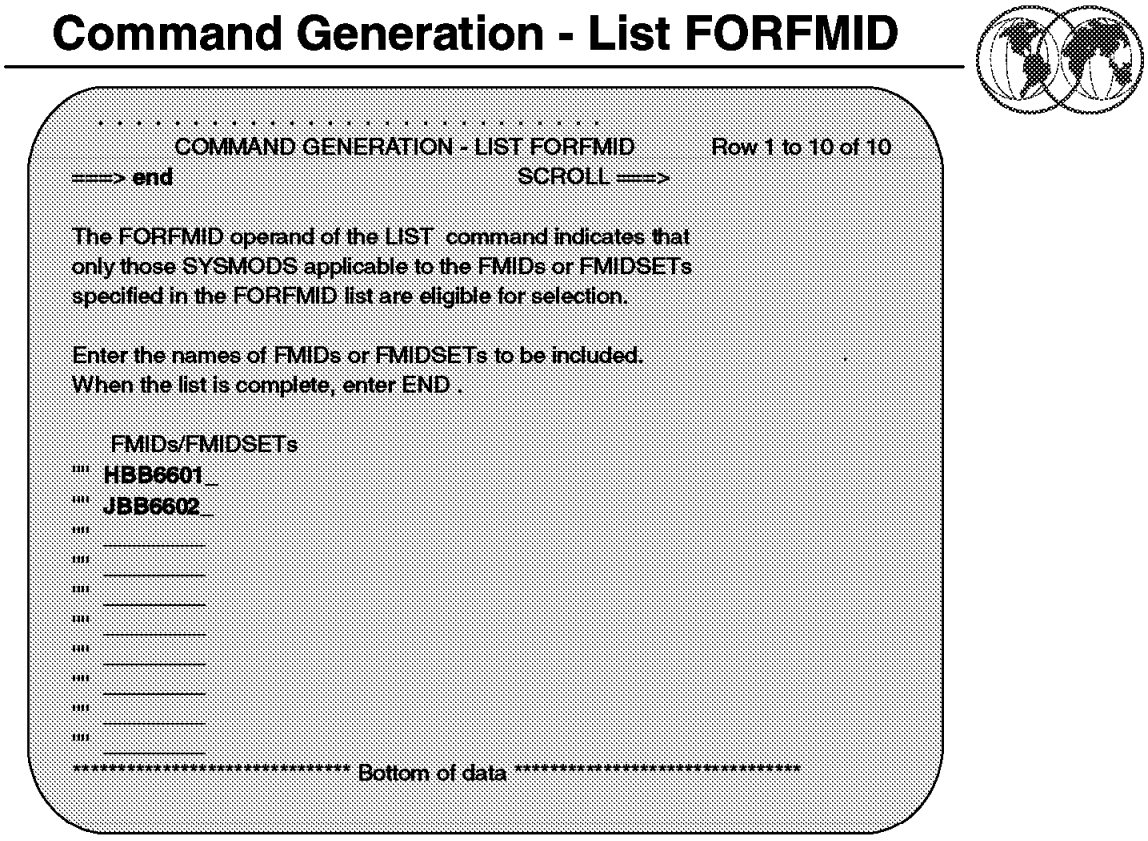


Figure 241. Command Generation - list FORFMID

3.12.10 Command Generation - list FORFMID

You can select one or more FMIDs in this panel. When you complete the list of names of FMIDs or FMIDSETs, Enter END on the command line.

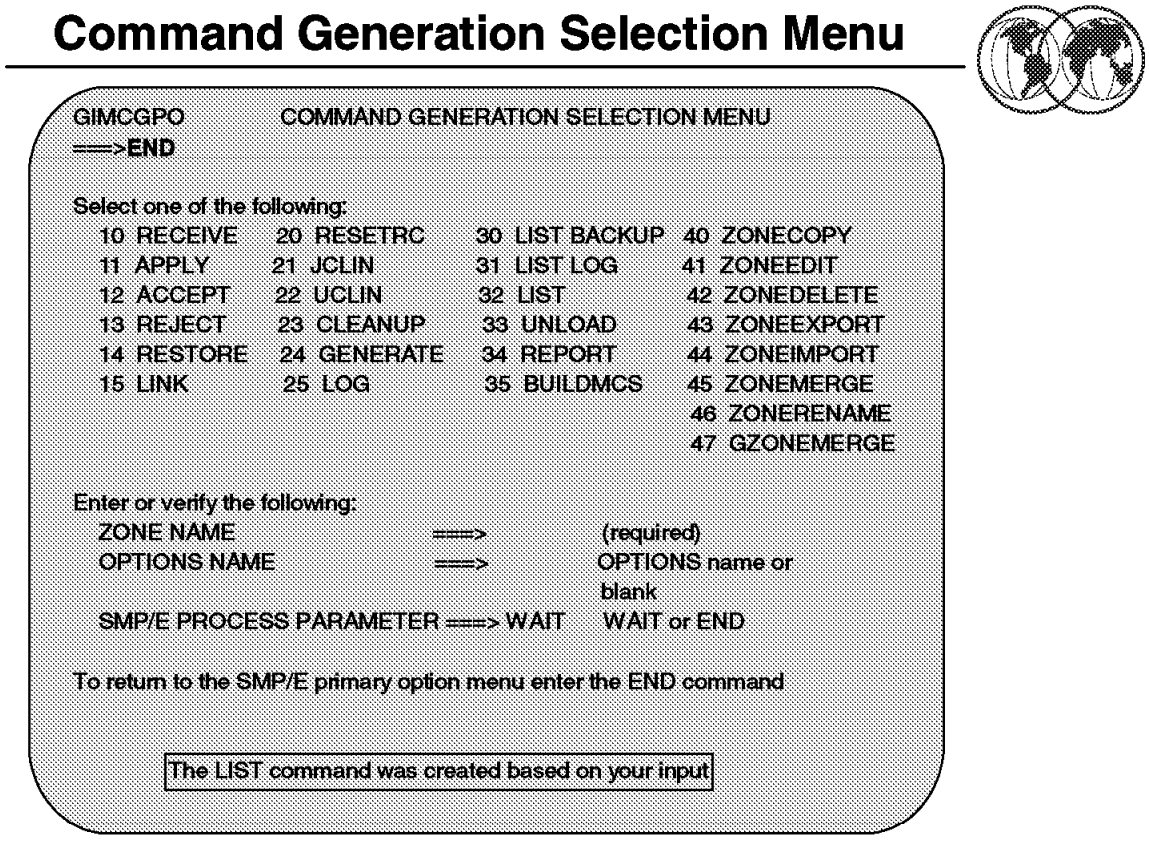


Figure 242. Command Generation Selection Menu

3.12.11 Command Generation Selection Menu

After you finish your selection, SMP/E will issue the message *The LIST command was created based on your input*, as shown in the visual.

Now, the job that you want is generated by SMP/E. If you want any other service of SMP/E, you can go again to the panel and select another command. The second command will be generated in the same job as the previous one. When you do not have any other changes to make, you can enter END in the command line.

Command Generation - SUBMIT

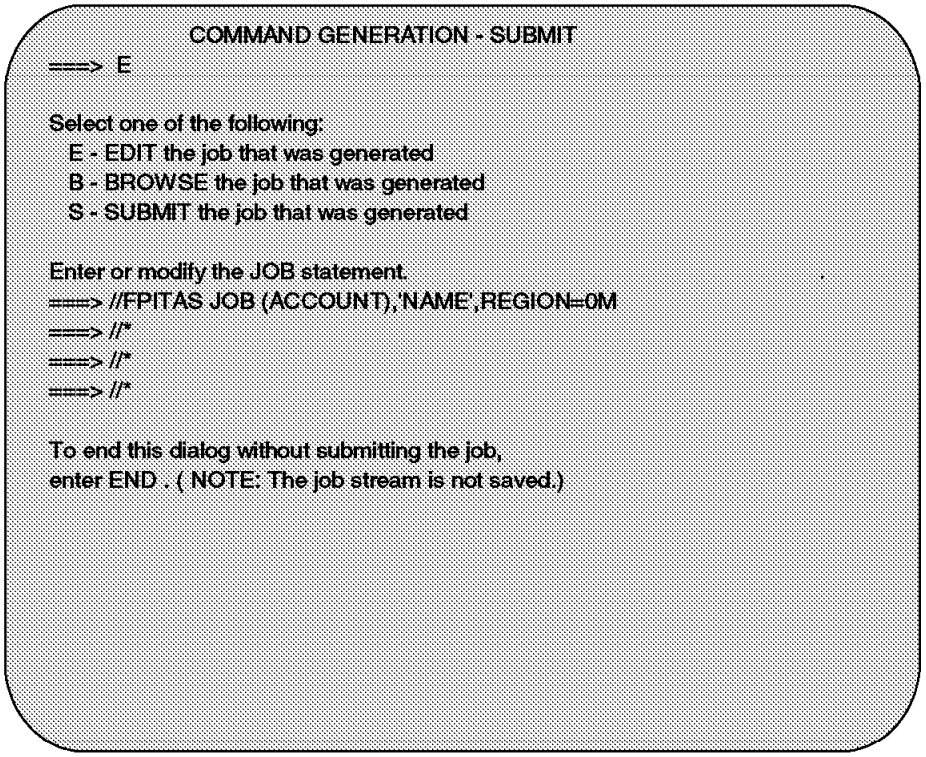


Figure 243. Command Generation - SUBMIT

3.12.12 Command Generation - SUBMIT

After Entering END on the previous visual, you receive the panel shown in this visual.

In this panel you can Edit the job (Option E) or submit the job (Option S). When you submit the job, the job executes and you can then browse the output to obtain the information you requested.

The Generated Job

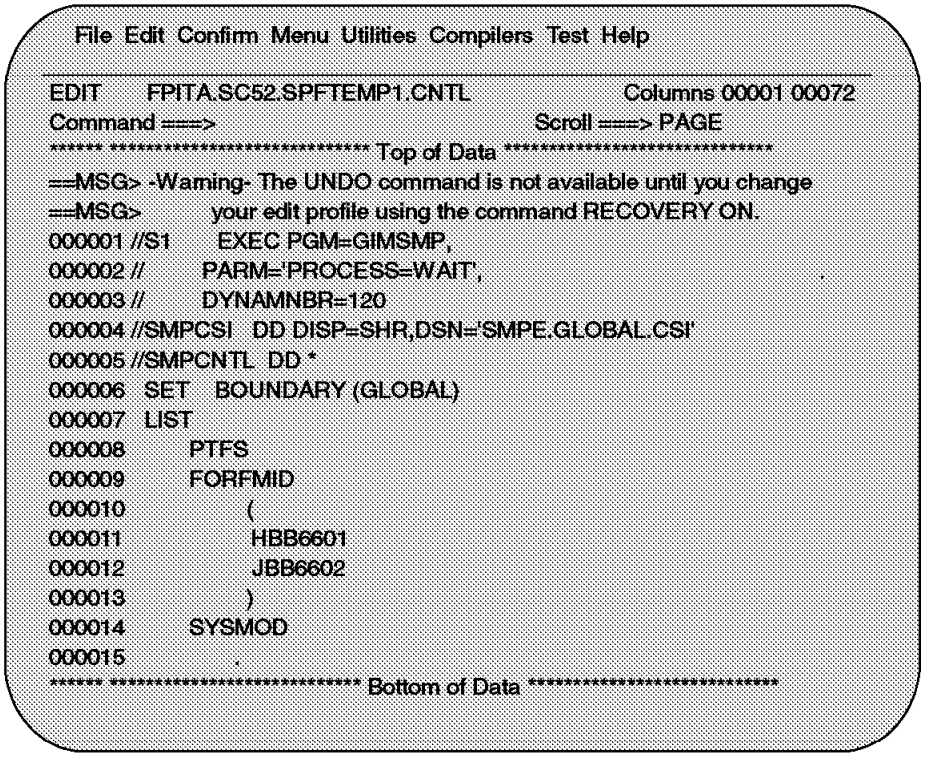


Figure 244. The generated job

3.12.13 The generated job

This is the job generated by SMP/E; you can submit it and it will execute the commands that you chose.

Appendix A. Special Notices

This publication is intended to help new system programmers who need to understand S/390 and the OS/390 operating system. The information in this publication is not intended as the specification of any programming interfaces that are provided by OS/390 Versions. See the PUBLICATIONS section of the IBM Programming Announcement for OS/390 Version 2 Release 8, Program Number 5647-A01 for more information about what publications are considered to be product documentation.

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 program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

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

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

You can reproduce a page in this document as a transparency, if that page has the copyright notice on it. The copyright notice must appear on each page being reproduced.

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

ACF/VTAM
Advanced Function Printing
AFP
AnyNet
CICS
CICS/ESA
CICS/MVS
DB2
DFSMS
DFSMS/MVS
DFSMSdfp
DFSMSdss
DFSMShsm
DFSMSrmm
DFSORT
ESCON
FICON
IBM
IMS
InfoPrint
Intelligent Printer Data Stream
IPDS
IP PrintWay
Language Environment
MVS (block letters)
MVS/DFP
NetSpool
NetView
OpenEdition
OS/390
Parallel Sysplex
PrintWay
RACF
RAMAC
RMF
S/390
S/390 Parallel Enterprise Server
Sysplex Timer
VTAM

The following terms are trademarks of other companies:

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjobenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

SET, SET Secure Electronic Transaction, and the SET logo are trademarks owned by Secure Electronic Transaction LLC.

UNIX is a registered trademark in the United States and other countries licensed exclusively through the Open Group.

Other company, product, and service names may be trademarks or service marks of others.

Appendix B. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

B.1 IBM Redbooks

For information on ordering these ITSO publications see “How to get IBM Redbooks” on page 339.

- *OS/390 Release 5 Implementation*, SG24-5151
- *OS/390 Release 4 Implementation*, SG24-2089
- *OS/390 Release 3 Implementation*, SG24-2067
- *OS/390 Release 2 Implementation*, SG24-4834
- cit.OS/390 Security Server 1999 Updates Technical Presentation Guide, SG24-5627
- *Security in OS/390-based TCP/IP Networks*, SG24-5383
- *Hierarchical File System Usage Guide*, SG24-5482
- *Enhanced Catalog Sharing and Management*, SG245594
- *Integrated Catalog Facility Backup and Recovery*, SG24-5644
- *OS/390 Version 2 Release 6 UNIX System Services Implementation and Customiztion*, SG24-5178
- *IBM S/390 FICON Implementation Guide*, SG24-5169
- *Exploiting S/390 Hardware Cryptography with Trusted Key Entry*, SG24-5455
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *Introduction to Storage Area Network SAN*, SG2-45470
- *TCP/IP in a Sysplex*, SG24-5235
- *SecureWay Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements*, SG24-5631
- *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 1: Configuration and Routing*, SG24-5227
- *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229
- *OS/390 Workload Manager Implementation and Exploitation*, SG24-5326
- *ADSM Server-to-Server Implementation and Operation*, SG24-5244
- *Stay Cool on OS/390: Installing Firewall Technology*, SG24-2046
- *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268
- *TCP/IP OpenEdition Implementation Guide*, SG24-2141
- *IMS/ESA Version 5 Performance Guide*, SG24-4637
- *Parallel Sysplex Configuration: Overview*, SG24-2075
- *Parallel Sysplex Configuation: Cookbook*, SG24-2076
- *Parallel Sysplex Config.: Connectivity*, SG24-2077

- *DFSMS Optimizer Presentation Guide Update*, SG24-4477
- *MVS Parallel Sysplex Capacity Planning*, SG24-4680
- *Getting the Most Out of a Parallel Sysplex*, SG24-2073
- *OS/390 eNetwork Communication Server TCP/IP Implementation Guide Volume 2*, SG24-5228

B.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

B.3 Other resources

These publications are also relevant as further information sources:

- *OS/390 Initialization and Tuning Guide*, SC28-1751
- *OS/390 Initialization and Tuning Reference*, SC28-1752
- *OS/390 Introduction and Release Guide*, GC28-1725
- *OS/390 MVS JCL User's Guide*, SC28-1758
- *OS/390 MVS JCL Reference*, GC28-1757
- *OS/390 MVS System Diagnosis: Tools and Service Aids*, LY28-1085, (available to IBM licensed customers only)
- *Interactive System Productivity Facility Getting Started*, SC34-4440
- *OS/390 Security Server (RACF) System Programmer's Guide*, SC28-1913
- *OS/390 TSO/E Customization*, SC28-1965
- *OS/390 TSO/E Primer*, GC28-1967
- *OS/390 TSO/E User's Guide*, SC28-1968
- *OS/390 SMP/E Reference*, SC28-1806
- *OS/390 SMP/E User's Guide*, SC28-1740
- *OS/390 SMP/E Commands*, SC28-1805
- *Standard Packaging Rules for MVS-Based Products*, SC23-3695
- *OS/390 MVS System Commands*, GC28-1781
- *OS/390 MVS IPCS Commands*, GC28-1754
- *OS/390 MVS IPCS User's Guide*, GC28-1756

- *DFSMS/MVS Using Data Sets*, SC26-4922
- *OS/390 Planning for Installation*, GC28-1726
- *OS/390 MVS System Data Sets Definition*, GC28-1782
- *ICKDSF R16 Refresh, User's Guide*, GC35-0033
- *OS/390 MVS System Management Facilities (SMF)*, GC28-1783
- *EREP V3R5 Reference*, GC35-0152
- *OS/390 JES2 Commands*, GC28-1790
- *OS/390 Hardware Configuration Definition User's Guide*, SC28-1848
- *DFSMS/MVS DFSMSdss Storage Administration Reference*, SC26-4929
- *IBM ServerPac for OS/390 Using the Installation Dialog*, SC28-1244
- *OS/390 Hardware Configuration Definition Planning*, GC28-1750
- *OS/390 MVS Using the Subsystem Interface*, SC28-1502
- *DFSMS/MVS Version 1 Release 4: Managing Catalogs*, SC26-4914
- *DFSMS/MVS Version 1 Release4: Access Method Services for Integrated Catalog Facility*, SC26-4906
- *DFSMS/MVS: DFSMShsm Implementation and Customization Guide*, SH21-1078
- *DFSMS/MVS Access Method Services for ICF Catalogs*, SC26-4500
- *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920
- *OS/390 eNetwork Communications Server: SNA Resource Definition Reference*, SC31-8565
- *OS/390 eNetwork Communications Server SNA Resource Definition Samples*, SC31-8566
- *OS/390 eNetwork Communications Server: SNA Operation*, SC31-8567
- *OS/390 V2R7.0 eNetwork CS IP Configuration*, SC31-8513
- *eNetwork Communications Server: IP User's Guide* GC31-8514
- *OS/390 UNIX System Services Planning*, SC28-1890
- *OS/390 TCP/IP OpenEdition: Configuration Guide* SC31-8304
- *OS/390 Open Systems Adapter Support Facility User's Guide*, SC28-1855.
- *OS/390 V2R6.0 MVS Planning: APPC/MVS Management*, GC28-1807
- *Print Services Facility for OS/390: Customization*, S544-5622
- *DFSMS/MVS Planning for Installation*, SC26-4919
- *DFSMS/MVS Implementing System-Managed Storage*, SC26-3123
- *DFSMS/MVS Remote Copy Administrator's Guide and Reference*, SC35-0169
- *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913
- *DFSMS/MVS DFSMSdfp Diagnosis Guide*, SY27-9605
- *DFSMS/MVS DFSMSdfp Advanced Services*, SC26-4921
- *DFSMS/MVS Using Magnetic Tapes*, SC26-4923
- *DFSMS/MVS Utilities*, SC26-4926

- *Service Level Reporter User's Guide: Reporting*, SH19-6530
- *DFSMS/MVS Object Access Method Application Programmer's Reference*, SC26-4917
- *DFSMS/MVS Object Access Method Planning, Installation, and Storage Administration Guide for Object Support*, SC26-4918
- *DFSORT Installation and Customization*, SC26-7041
- *DFSORT Getting Started with DFSORT R14*, SC26-4109
- *DFSMS/MVS Network File System Customization and Operation*, SC26-7029
- *DFSMS Optimizer User's Guide and Reference*, SC26-7047
- *DFSMS/MVS DFSMSdss Storage Administration Guide*, SC26-4930
- *DFSMShsm Storage Administration Guide*, SH21-1076
- *DFSMShsm Storage Administration Reference*, SH21-1075
- *DFSMS/MVS Network File System User's Guide*, SC26-7028
- *DFSMS/MVS DFSMSrmm Guide and Reference*, SC26-4931
- *DFSMS/MVS DFSMSrmm Implementation and Customization Guide*, SC26-4932
- *MVS/ESA Storage Management Library Managing Data*, SC26-3124
- *MVS/ESA Storage Management Library Managing Storage Groups*, SC26-3125
- *MVS/ESA Storage Management Library Leading a Storage Administration Group*, SC26-3126.
- *DFSMS/MVS Using the Interactive Storage Management Facility*, SC26-4911
- *ADSTAR Distributed Storage Manager for MVS Administrator's Guide*, GC35-0277
- *OS/390 MVS Programming: Assembler Services Guide*, GC28-1762
- *OS/390 MVS Programming: Resource Recovery*, GC28-1739
- *OS/390 MVS Setting Up a Sysplex*, GC28-1779
- *OS/390 MVS Sysplex Services Guide*, GC28-1771
- *OS/390 Parallel Sysplex Systems Management*, GC28-1861
- *OS/390 MVS Systems Codes*, GC28-1780
- *OS/390 MVS System Messages Volume 1*, GC28-1784
- *OS/390 MVS System Messages Volume 2*, GC28-1785
- *OS/390 MVS System Messages Volume 3*, GC28-1786
- *OS/390 MVS System Messages Volume 4*, GC28-1787
- *OS/390 MVS System Messages Volume 5*, GC28-1788
- *OS/390 MVS Installation Exits*, SC28-1753
- *OS/390 MVS Diagnosis Reference*, SY28-1084
- *CICS User's Handbook*, SX33-1188
- *CICS Diagnosis Guide*, LX33-6093
- *MQSeries for MVS/ESA Messages and Codes*, GC33-0819

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the redbook fax order form to:

In United States:	e-mail address: usib6fpl@ibmmail.com
Outside North America:	Contact information is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

A

abend. Termination of a task before its completion because of an error condition that cannot be resolved by recovery facilities while the task executing.

ACB. Access method control block.

access. A specific type of interaction between a subject and an object that results in the flow of information from one to the other.

access authority. An authority that relates to a request for a type of access to protected resources. In RACF, the access authorities are NONE, READ, UPDATE, ALTER, and EXECUTE.

access list. A list within a profile of all authorized users and their access authorities.

access method control block (ACB).. A control block that links an application program to VTAM.

ACDS. Active control data set.

ACF/VTAM. An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability. VTAM runs under MVS (OS/VS1 and OS/VS2), VSE, and VM/SP and supports direct control application programs and subsystems such as VSE/POWER.

ACIF. (1) AFP conversion and indexing facility. (2) A PSF utility program that converts a print file into AFP, MO:DCA-P, creates an index file for input data, and collects resources used by an AFP document into separate file.

action message retention facility (AMRF). A facility that, when active, retains all action messages except those specified by the installation in the MPFLSTxx member in effect.

action message sequence number. A decimal number assigned to action messages.

Advanced Function Presentation (AFP). A set of licensed programs, together with user applications, that use the all-points-addressable concept to print on presentation devices. AFP includes creating, formatting, archiving, retrieving, viewing, distributing, and printing information.

Advanced Program-to-Program Communications (APPC). A set of inter-program communication services that support cooperative transaction processing in a SNA network.

AFP. Advanced Function Presentation.

AFP Printer Driver for Windows. A component of Infoprint Server for OS/390 that runs on a Windows 95 or Windows NT workstation and creates output in AFP format, for printing on AFP printers.

AFP Viewer plug-in for Windows. A component of Infoprint Server for OS/390 that runs on a Windows 95 or Windows NT workstation and allows you to view files in AFP format.

AIX operating system. IBM′s implementation of the UNIX operating system. The RS/6000 system, among others, runs the AIX operating system.

allocate. To assign a resource for use in performing a specific task.

alphanumeric character. A letter or a number.

amode. Addressing mode. A program attribute that can be specified (or defaulted) for each CSECT, load module, and load module alias. AMODE states the addressing mode that is expected to be in effect when the program is entered.

AMRF. action message retention facility

AOR. Application-owning region

APPC. Advanced Program-to-Program Communications

APPN. Advanced Peer-to-Peer Networking.

ASCII (American Standard Code for Information Interchange). The standard code, using a coded character set consisting of 7-bit coded characters (8-bit including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

audit. To review and examine the activities of a data processing system mainly to test the adequacy and effectiveness of procedures for data security and data accuracy.

authority. The right to access objects, resources, or functions.

authorization checking. The action of determining whether a user is permitted access to a RACF-protected resource.

Authorized Program Analysis Report (APAR). A request for correction of problem caused by a defect in a current unaltered release of a program.

authorized program facility (APF). A facility that permits identification of programs authorized to use restricted functions.

automated operations. Automated procedures to replace or simplify actions of operators in both systems and network operations.

AVR. Automatic volume recognition.

B

banner page. A page printed before the data set is printed.

basic mode. A central processor mode that does not use logical partitioning. Contrast with logically partitioned (LPAR) mode.

batch message processing (BMP) program. An IMS batch processing program that has access to online databases and message queues. BMPs run online, but like programs in a batch environment, they are started with job control language (JCL).

batch-oriented BMP program. A BMP program that has access to online databases and message queues while performing batch-type processing. A batch-oriented BMP does not access the IMS message queues for input or output. It can access online databases, GSAM databases, and MVS files for both input and output.

BMP. Batch message processing (BMP) program.

broadcast. (1) Transmission of the same data to all destinations. (2) Simultaneous transmission of data to more than one destination.

binary data. (1) Any data not intended for direct human reading. Binary data may contain unprintable characters, outside the range of text characters. (2) A type of data consisting of numeric values stored in bit patterns of 0s and 1s. Binary data can cause a large number to be placed in a smaller space of storage.

BIND. In SNA, a request to activate a session between two logical units (LUs).

buffer. A portion of storage used to hold input or output data temporarily.

buffered device. A device where the data is written to a hardware buffer in the device before it is placed on the paper (for example, IBM 3820).

burst. To separate continuous-forms paper into single sheets.

C

cache structure. A coupling facility structure that enables high-performance sharing of cached data by multisystem applications in a sysplex. Applications can use a cache structure to implement several different types of caching systems, including a store-through or a store-in cache.

carriage control character. An optional character in an input data record that specifies a write, space, or skip operation.

carriage return (CR). (1) A keystroke generally indicating the end of a command line. (2) In text data, the action that indicates to continue printing at the left margin of the next line. (3) A character that will cause printing to start at the beginning of the same physical line in which the carriage return occurred.

CART. Command and response token.

case-sensitive. Pertaining to the ability to distinguish between uppercase and lowercase letters.

catalog. (1) A directory of files and libraries, with reference to their locations. (2) To enter information about a file or a library into a (3) The collection of all data set indexes that are used by the control program to locate a volume containing a specific data set.

CBPDO. Custom Built Product Delivery Offering.

CEC. Synonym for central processor complex (CPC).

central processor (CP). The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

central processor complex (CPC). A physical collection of hardware that includes main storage, one or more central processors, timers, and channels.

CFRM. Coupling facility resource management.

channel-to-channel (CTC). Refers to the communication (transfer of data between programs on opposite sides of a channel-to-channel adapter (CTCA

channel-to-channel adapter (CTCA). An input/output device that is used a program in one system to communicate with a program in another system.

checkpoint. (1) A place in a routine where a check, or a recording of data for restart purposes, is performed. (2) A point at which information about the status of a job and the system can be recorded so that the job step can be restarted later.

checkpoint write. Any write to the checkpoint data set. A general term for the primary, intermediate, and final writes that update any checkpoint data set.

CICS. Customer Information Control System.

CICSplex. A group of connected CICS regions.

CICSPlex SM. CICSPlex System Manager

client. A functional unit that receives shared services from a server. See also client-server.

client-server. In TCP/IP, the model of interaction in distributed data processing in which a program at one site sends a request to a program at another site and awaits a response. The requesting program is called a client; the answering program is called a server.

CMOS. Complementary metal-oxide semiconductor.

CNGRPxx. The SYS1.PARMLIB member that defines console groups for the system or sysplex.

code page. (1) A table showing codes assigned to character sets. (2) An assignment of graphic characters and control function meanings to all code points. (3) Arrays of code points representing characters that establish ordinal sequence (numeric order) of characters. (4) A particular assignment of hexadecimal identifiers to graphic elements.

code point. A 1-byte code representing one of 256 potential characters.

coexistence. Two or more systems at different levels (for example, software, service or operational levels) that share resources. Coexistence includes the ability of a system to respond in the following ways to a new function that was introduced on another system with which it shares resources: ignore a new function, terminate gracefully, support a new function.

command and response token (CART). A parameter on WTO, WTOR, MGCRE, and certain TSO/E commands and REXX execs that allows you to link commands and their associated message responses.

command prefix facility (CPF). An MVS facility that allows you to define and control subsystem and other command prefixes for use in a sysplex.

COMMDS. Communications data set.

complementary metal-oxide semiconductor (CMOS). A technology that combines the electrical properties of positive and negative voltage requirements to use considerably less power than other types of semiconductors.

connection. In TCP/IP, the path between two protocol applications that provides reliable data stream delivery service. In Internet communications, a connection extends from a TCP application on one syste system to a TCP application on another system.

console. That part of a computer used for communication between the operator or user and the computer.

console group. In MVS, a group of consoles defined in CNGRPxx, each of whose members can serve as an alternate console in console or hardcopy recovery or as a console to display synchronous messages.

CONSOLxx. The SYS1.PARMLIB member used to define message handling, command processing, and MCS consoles.

control unit. Synonymous with device control unit.

conversation. A logical connection between two programs over an LU type 6.2 session that allows them to communicate with each other while processing a transaction.

conversational. Pertaining to a program or a system that carries on a dialog with a terminal user, alternately accepting input and then responding to the input quickly enough for the user to maintain a train of thought.

copy group. One or more copies of a page of paper. Each copy can have modifications, such as text suppression, page position, forms flash, and overlays.

couple data set. A data set that is created through the XCF couple data set format utility and, depending on its designated type, is shared by some or all of the MVS systems in a sysplex. See also sysplex couple data set.

coupling facility. A special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex.

coupling facility channel.. A high bandwidth fiber optic channel that provides the high-speed connectivity required for data sharing between a coupling facility and the central processor complexes directly attached to it.

coupling services. In a sysplex, the functions of XCF that transfer data and status between members of a group residing on one or more MVS systems in the sysplex.

CP. Central processor.

CPC. Central processor complex.

CPF. Command prefix facility.

cross-system coupling facility (XCF). XCF is a component of MVS that provides functions to support cooperation between authorized programs running within a sysplex.

cryptography. The transformation of data to conceal its meaning.

cryptographic key. A parameter that determines cryptographic transformations between plaintext and ciphertext.

CTC. Channel-to-channel.

Customer Information Control System (CICS). An IBM licensed program tha that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

D

DAE. Dump analysis and elimination.

daemon. A program that runs unattended to perform a standard service.

DASD. Direct access storage device.

data definition name. The name of a data definition (DD) statement, which corresponds to a data control block that contains the same name. Abbreviated as ddname.

data definition (DD) statement. A job control statement that describes a data set associated with a particular job step.

data integrity. The condition that exists as long as accidental or intentional destruction, alteration, or loss of data does not occur.

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

data set label. (1) A collection of information that describes the attributes of a data set and is normally stored on the same volume as the data set. (2) A general term for data set control blocks and tape data set labels.

data set separator pages. Those pages of printed output that delimit data sets.

data sharing. The ability of concurrent subsystems (such as DB2 or IMS DB) or application programs to directly access and change the same data while maintaining data integrity.

data stream. (1) All information (data and control commands) sent over a data link usually in a single read or write operation. (2) A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format.

DBCS. Double-byte character set.

DBCTL. IMS Database Control.

DBRC. Database Recovery Control.

DB2. DATABASE 2 for MVS/ESA.

DB2 data sharing group. A collection of one or more concurrent DB2 subsystems that directly access and change the same data while maintaining data integrity.

DB2 PM. DB2 Performance Monitor.

deallocate. To release a resource that is assigned to a specific task.

default. A value, attribute, or option that is assumed when no alternative is specified by the user.

destination node. The node that provides application services to an authorized external user.

device control unit. A hardware device that controls the reading, writing, or displaying of data at one or more input/output devices or terminals.

device number. The unique number assigned to an external device.

device type. The general name for a kind of device; for example, 3330.

DFSMS. Data Facility Storage Management Subsystem.

direct access storage device (DASD). A device in which the access time effectively independent of the location of the data.

directory. (1) A type of file containing the names and controlling information for other files or other directories. Directories can also contain subdirectories, which can contain subdirectories of their own. (2) A file that contains directory entries. No two directory entries in the same directory can have the same name. (POSIX.1). (3) A file that points to files and to other directories. (4) An index used by a control program to locate blocks of data that are stored in separate areas of a data set in direct access storage.

display console. In MVS, an MCS console whose input/output function you can control.

DLL filter. A filter that provides one or more of these functions in a dynamic load library - init(), prolog(), process(), epilog(), and term(). See cfilter.h and cfilter.c in the /usr/lpp/Printsrv/samples/ directory for more information. See also filter. Contrast with DLL filter.

DOM. An MVS macro that removes outstanding WTORs or action messages that have been queued to a console end-of-tape-marker. A marker on a

magnetic tape used to indicate the end of the permissible recording area, for example, a photo-reflective strip a transparent section of tape, or a particular bit pattern.

dotted decimal notation. The syntactical representation for a 32-bit integer that consists of four 8-bit numbers written in base 10 with periods (dots) separating them. It is used to represent IP addresses.

double-byte character set (DBCS). A set of characters in which each character is represented by a two-bytes code. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires two bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with single-byte character set.

drain. Allowing a printer to complete its current work before stopping the device.

E

entry area. In MVS, the part of a console screen where operators can enter commands or command responses.

EMIF. ESCON Multiple Image Facility.

Enterprise Systems Connection (ESCON). A set of products and services that provides a dynamically connected environment using optical cables as a transmission medium.

EPDM. IBM SystemView Enterprise Performance Data Manager/MVS.

ESCD. ESCON Director.

ESCM. ESCON Manager. The licensed program System Automation for OS/390 includes all of the function previosuly provided by ESCM.

ESCON. Enterprise Systems Connection.

ETR. External Time Reference. See also Sysplex Timer.

extended MCS console. In MVS, a console other than an MCS console from which operators or programs can issue MVS commands and receive messages. An extended MCS console is defined through an OPERPARM segment.

F

FMID. Function modification identifier. The IBM release-specific product identifier such as HJE6610 for OS/390 Release 1 JES2.

FOR. File-owning region.

frame. For a System/390 microprocessor cluster, a frame contains one or two central processor complexes (CPCs), support elements, and AC power distribution.

FSS. functional subsystem. An address space uniquely identified as performing a specific function related to the JES. An example of an FSS is the program Print Services Facility that operates the 3800 Model 3 an 38xx printers.

functional subsystem (FSS). An address space uniquely identified as performing a specific function related to the JES.

functional subsystem application (FSA). The functional application program managed by the functional subsystem.

functional subsystem interface (FSI). The interface through which JES2 JES3 communicate with the functional subsystem.

G

gateway node. A node that is an interface between networks.

generalized trace facility (GTF). Like system trace, gathers information used to determine and diagnose problems that occur during system operation. Unlike system trace, however, GTF can be tailored to record very specific system and user program events.

global access checking. The ability to allow an installation to establish an in-storage table of default values for authorization levels for selected resources.

global resource serialization. A function that provides an MVS serialization mechanism for resources (typically data sets) across multiple MVS images.

global resource serialization complex. One or more MVS systems that use global resource serialization to serialize access to shared resources (such as data sets on shared DASD volumes).

group. A collection of RACF users who can share access authorities for protected resources.

GTF. Generalized trace facility.

H

hardcopy log. In systems with multiple console support or a graphic console, a permanent record of system activity.

hardware. Physical equipment, as opposed to the computer program or method of use; for example, mechanical, magnetic, electrical, or electronic devices. Contrast with software.

hardware configuration dialog. In MVS, a panel program that is part of the hardware configuration definition. The program allows an installation to define devices for MVS system configurations.

Hardware Management Console. A console used to monitor and control hardware such as the System/390 microprocessors.

HCD. Hardware Configuration Definition.

highly parallel. Refers to multiple systems operating in parallel, each of which can have multiple processors. See also n-way.

I

ICMF. Integrated Coupling Migration Facility.

IMS. Information Management System.

IMS DB. Information Management System Database Manager.

IMS DB data sharing group. A collection of one or more concurrent IMS DB subsystems that directly access and change the same data while maintaining data integrity.

IMS TM. Information Management System Transaction Manager.

initial program load (IPL). The initialization procedure that causes an operating system to begin operation.

instruction line. In MVS, the part of the console screen that contains messages about console control and input errors.

internal reader. A facility that transfers jobs to the job entry subsystem (JES2 or JES3).

IOCDS. Input/output configuration data set.

IOCP. Input/output configuration program.

IODF. Input/output definition file.

IPL. Initial program load.

IRLM. Internal resource lock manager.

ISPF. Interactive System Productivity Facility.

J

JES common coupling services. A set of macro-driven services that provide the communication interface between JES members of a sysplex. Synonymous with JES XCF.

JESXCF. JES cross-system coupling services. The MVS component, common to both JES2 and JES3, that provides the cross-system coupling services to either JES2 multi-access spool members or JES3 complex members, respectively.

JES2. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In an installation with more than one processor, each JES2 processor independently controls its job input, scheduling, and output processing.

JES2 multi-access spool configuration. A multiple MVS system environment that consists of two or more JES2 processors sharing the same job queue and spool

JES3. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In complexes that have several loosely-coupled processing units, the JES3 program manages processors so that the global processor exercises centralized control over the local processors and distributes jobs to them via a common job queue.

JES3 complex. A multiple MVS system environment that allows JES3 subsystem consoles and MCS consoles with a logical association to JES3 to receive messages and send commands across systems.

job entry subsystem (JES). A system facility for spooling, job queuing, and managing the scheduler work area.

job separator page data area (JSPA). A data area that contains job-level information for a data set. This information is used to generate job header, job trailer or data set header pages. The JSPA can be used by an installation-defined JES2 exit routine to duplicate the information currently in the JES2 separator page exit routine.

job separator pages. Those pages of printed output that delimit jobs.

K

keyword. A part of a command operand or SYS1.PARMLIB statement that consists of a specific character string (such as NAME= on the CONSOLE statement of CONSOLxx).

L

LIC. Licensed Internal Code.

list structure. A coupling facility structure that enables multisystem applications in a sysplex to share information organized as a set of lists or queues. A list structure consists of a set of lists and an optional lock table, which can be used for serializing resources in the list structure. Each list consists of a queue of list entries.

lock structure. A coupling facility structure that enables applications in a sysplex to implement customized locking protocols for serialization of application-defined resources. The lock structure supports shared, exclusive, and application-defined lock states, as well as generalized contention management and recovery protocols.

logical partition (LP). A subset of the processor hardware that is defined to support an operating system. See also logically partitioned (LPAR) mode.

logically partitioned (LPAR) mode. A central processor complex (CPC) power-on reset mode that enables use of the PR/SM feature and allows an operator to allocate CPC hardware resources (including central processors, central storage, expanded storage, and channel paths) among logical partitions. Contrast with basic mode.

logical unit (LU). In SNA, a port through which an end user accesses th SNA network in order to communicate with another end user and through which the end user accesses the functions provided by system services control points (SSCPs).

logical unit type 6.2. The SNA logical unit type that supports general communication between programs in a cooperative processing environment.

loosely coupled. A multisystem structure that requires a low degree of interaction and cooperation between multiple MVS images to process a workload. See also tightly coupled.

LP. Logical partition.

LPAR. Logically partitioned (mode).

M

MAS. Multi-access spool.

master console. In an MVS system or sysplex, the main console used for communication between the operator and the system from which all MVS commands can be entered. The first active console with AUTH(MASTER) defined becomes the master console in a system or sysplex.

master console authority. In a system or sysplex, a console defined with AUTH(MASTER) other than the master console from which all MVS commands can be entered.

master trace. A centralized data tracing facility of the master scheduler, used in servicing the message processing portions of MVS.

MCS. Multiple console support.

MCS console. A non-SNA device defined to MVS that is locally attached to an MVS system and is used to enter commands and receive messages.

member. A specific function (one or more modules/routines) of a multisystem application that is defined to XCF and assigned to a group by the multisystem application. A member resides on one system in the sysplex and can use XCF services to communicate (send and receive data) with other members of the same group.

message processing facility (MPF). A facility used to control message retention, suppression, and presentation.

message queue. A queue of messages that are waiting to be processed or waiting to be sent to a terminal.

message text. The part of a message consisting of the actual information that is routed to a user at a terminal or to a program.

microprocessor. A processor implemented on one or a small number of chips.

mixed complex. A global resource serialization complex in which one or more of the systems in the global resource serialization complex are not part of a multisystem sysplex.

MP. Multiprocessor.

MPF. Message processing facility.

MPFLSTxx. The SYS1.PARMLIB member that controls the message processing facility for the system.

MRO. Multiregion operation.

multiple console support (MCS). The operator interface in an MVS system.

multi-access spool (MAS). A complex of multiple processors running MVS/JES2 that share a common JES2 spool and JES2 checkpoint data set.

multiprocessing. The simultaneous execution of two or more computer programs or sequences of instructions. See also parallel processing.

multiprocessor (MP). A CPC that can be physically partitioned to form two operating processor complexes.

multisystem application. An application program that has various functions distributed across MVS images in a multisystem environment.

multisystem console support. Multiple console support for more than one system in a sysplex. Multisystem console support allows consoles on different systems in the sysplex to communicate with each other (send messages and receive commands)

multisystem environment. An environment in which two or more MVS images reside in one or more processors, and programs on one image can communicate with programs on the other images.

multisystem sysplex. A sysplex in which two or more MVS images are allowed to be initialized as part of the sysplex.

MVS image. A single occurrence of the MVS/ESA operating system that has the ability to process work.

MVS router. The MVS router is a system service that provides an installation with centralized control over system security processing.

MVS system. An MVS image together with its associated hardware, which collectively are often referred to simply as a system, or MVS system.

MVS/ESA. Multiple Virtual Storage/ESA.

MVSCP. MVS configuration program.

N

n-way. The number (n) of CPs in a CPC. For example, a 6-way CPC contains six CPs.

NIP. Nucleus initialization program.

NJE. Network job entry.

no-consoles condition. A condition in which the system is unable to access any full-capability console device.

nonstandard labels. Labels that do not conform to American National Standard or IBM System/370 standard label conventions.

nucleus initialization program (NIP). The stage of MVS that initializes the control program; it allows the operator to request last minute changes to certain options specified during initialization.

O

offline. Pertaining to equipment or devices not under control of the processor.

OLTP. Online transaction processing.

online. Pertaining to equipment or devices under control of the processor.

OPC/ESA. Operations Planning and Control.

operating system (OS). Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible.

operations log. In MVS, the operations log is a central record of communications and system problems for each system in a sysplex.

OPERLOG. The operations log.

OPERPARM. In MVS, a segment that contains information about console attributes for extended MCS consoles running on TSO/E.

OS/390. OS/390 is a network computing-ready, integrated operating system consisting of more than 50 base elements and integrated optional features delivered as a configured, tested system.

OS/390 Network File System. A base element of OS/390, that allows remote access to MVS host processor data from workstations, personal computers, or any other system on a TCP/IP network that is using client software for the Network File System protocol.

OS/390 UNIX System Services (OS/390 UNIX). The set of functions provided by the SHELL and UTILITIES, kernel, debugger, file system, C/C++ Run-Time Library, Language Environment, and other elements of the OS/390 operating system that allow users to write and run application programs that conform to UNIX standards.

P

parallel processing. The simultaneous processing of units of work by many servers. The units of work can be either transactions or subdivisions of large units of work (batch). See also highly parallel.

Parallel Sysplex. A sysplex that uses one or more coupling facilities.

partitionable CPC. A CPC that can be divided into 2 independent CPCs. See also physical partition, single-image mode, MP, side.

partitioned data set (PDS). A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE). A system-managed data set that contains 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.

password. A unique string of characters known to a computer system and to a user, who must specify the character string to gain access to a system and to the information stored within it.

permanent data set. A user-named data set that is normally retained for longer than the duration of a job or interactive session. Contrast with temporary data set.

PFK. Program function key.

PFK capability. On a display console, indicates that program function keys are supported and were specified at system generation.

PFKTABxx. The SYS1.PARMLIB member that controls the PFK table settings for MCS consoles in a system.

physical partition. Part of a CPC that operates as a CPC in its own right, with its own copy of the operating system.

physically partitioned (PP) configuration. A system configuration that allows the processor controller to use both central processor complex (CPC) sides as individual CPCs. The A-side of the processor controller controls side 0; the B-side of the processor controller controls side 1. Contrast with single-image (SI) configuration.

PR/SM. Processor Resource/Systems Manager.

Print Services Facility (PSF). The access method that supports the 3800 Printing Subsystem Models 3 and 8. PSF can interface either directly to a user′s application program or indirectly through the Job Entry Subsystem (JES) of MVS.

printer. (1) A device that writes output data from a system on paper or other media.

processor controller. Hardware that provides support and diagnostic functions for the central processors.

Processor Resource/Systems Manager (PR/SM). The feature that allows the processor to use several MVS images simultaneously and provides logical partitioning capability. See also LPAR.

profile. Data that describes the significant characteristics of a user, a group of users, or one or more computer resources.

program function key (PFK). A key on the keyboard of a display device that passes a signal to a program to call for a particular program operation.

program status word (PSW). A doubleword in main storage used to control the order in which instructions are executed, and to hold and indicate the status of the computing system in relation to a particular program.

pseudo-master console. A subsystem-allocatable console that has system command authority like that of an MCS master console.

PSW. Program status word.

R

RACF. See Resource Access Control Facility.

RAID. See redundant array of independent disk.

RAMAC Virtual Array (RVA) system. An online, random access disk array storage system composed of disk storage and control unit combined into a single frame.

read access. Permission to read information.

recording format. For a tape volume, the format of the data on the tape, for example, 18, 36, 128, or 256 tracks.

recovery. The process of rebuilding data after it has been damaged or destroyed, often by using a backup copy of the data or by reapplying transactions recorded in a log.

redundant array of independent disk (RAID). A disk subsystem architecture that combines two or more physical disk storage devices into a single logical device to achieve data redundancy.

remote operations. Operation of remote sites from a host system.

Resource Access Control Facility (RACF). An IBM-licensed program or a base element of OS/390, that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system and logging the detected accesses to protected resources.

restructured extended executor (REXX). A general-purpose, procedural language for end-user personal programming, designed for ease by both casual general users and computer professionals. It is also useful for application macros. REXX includes the capability of issuing commands to the underlying operating system from these macros and procedures. Features include powerful character-string manipulation, automatic data typing, manipulation of objects familiar to people, such as words, numbers, and names, and built-in interactive debugging.

REXX. See restructured extended executor.

RMF. Resource Measurement Facility.

rmode. Residency mode. A program attribute that can be specified (or defaulted) for each CSECT, load module, and load module alias. RMODE states the virtual storage location (either above 16 megabytes or anywhere in virtual storage) where the program should reside.

roll mode. The MCS console display mode that allows messages to roll of off the screen when a specified time interval elapses.

roll-deletable mode. The console display mode that allows messages to roll off the screen when a specified time interval elapses. Action messages remain at the top of the screen where operators can delete them.

routing. The assignment of the communications path by which a message will reach its destination.

routing code. A code assigned to an operator message and used to route the message to the proper console.

RVA. See RAMAC Virtual Array system.

S

SCDS. Source control data set.

SDSF. System Display and Search Facility.

shared DASD option. An option that enables independently operating computing systems to jointly use common data residing on shared direct access storage devices.

side. A part of a partitionable CPC that can run as a physical partition and is typically referred to as the A-side or the B-side.

single point of control. The characteristic a sysplex displays when you can accomplish a given set of tasks from a single workstation, even if you need multiple IBM and vendor products to accomplish that particular set of tasks.

single system image. The characteristic a product displays when multiple images of the product can be viewed and managed as one image.

single-image (SI) mode. A mode of operation for a multiprocessor (MP) system that allows it to function as one CPC. By definition, a uniprocessor (UP) operates in single-image mode. Contrast with physically partitioned (PP) configuration.

single-system sysplex. A sysplex in which only one MVS system is allowed to be initialized as part of the sysplex. In a single-system sysplex, XCF provides XCF services on the system but does not provide signalling services between MVS systems. See also multisystem sysplex, XCF-local mode.

SLR. Service Level Reporter.

small computer system interface (SCSI). A standard hardware interface that enables a variety of peripheral devices to communicate with one another.

SMF. System management facilities.

SMP/E. System Modification Program Extended.

SMS. Storage Management Subsystem.

SMS communication data set. The primary means of communication among systems governed by a single SMS configuration. The SMS communication data set (COMMDS) is a VSAM linear data set that contains the current utilization statistics for each system-managed volume, which SMS uses to help balance space usage among systems.

SMS configuration. The SMS definitions and routines that the Storage Management Subsystem uses to manage storage.

SMS system group. All systems in a sysplex that share the same SMS configuration and communications data sets, minus any systems in the sysplex that are defined individually in the SMS configuration.

software. (1) All or part of the programs, procedures, rules, and associated documentation of a data processing system. (2) Contrast with hardware. A set of programs, procedures, and, possibly, associated documentation concerned with the operation of a data processing system. For example, compilers, library

routines, manuals, circuit diagrams. Contrast with hardware.

spanned record. A logical record contained in more than one block.

status-display console. An MCS console that can receive displays of system status but from which an operator cannot enter commands.

storage administrator. A person in the data processing center who is responsible for defining, implementing, and maintaining storage manageme policies.

storage class. A collection of storage attributes that identify performance goals and availability requirements, defined by the storage administrator, used to select a device that can meet those goals and requirements.

storage group. A collection of storage volumes and attributes, defined the storage administrator. The collections can be a group of DASD volume or tape volumes, or a group of DASD, optical, or tape volumes treated as single object storage hierarchy. See tape storage group.

storage management. The activities of data set allocation, placement, monitoring, migration, backup, recall, recovery, and deletion. These can be done either manually or by using automated processes. cThe Storage Management Subsystem automates these processes for you, while optimizing storage resources. See also Storage Management Subsystem.

Storage Management Subsystem (SMS). A DFSMS/MVS 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.

storage subsystem. A storage control and its attached storage devices. See also tape subsystem.

structure. A construct used by MVS to map and manage storage on a coupling facility. See cache structure, list structure, and lock structure.

subsystem-allocatable console. A console managed by a subsystem like JES3 or NetView used to communicate with an MVS system.

subsystem interface (SSI). An MVS component that provides communication between MVS and JES.

supervisor call instruction (SVC). An instruction that interrupts a program being executed and passes control to the supervisor so that it can perform a specific service indicated by the instruction.

support element. A hardware unit that provides communications, monitoring, and diagnostic functions to a central processor complex (CPC).

SVC routine. A control program routine that performs or begins a contro program service specified by a supervisor call instruction.

symmetry. The characteristic of a sysplex where all systems, or certain subsets of the systems, have the same hardware and software configurations and share the same resources.

synchronous messages. WTO or WTOR messages issued by an MVS system during certain recovery situations.

SYSLOG. The system log data set.

sysplex. A set of MVS systems communicating and cooperating with each other through certain multisystem hardware components and software services to process customer workloads. See also MVS system, Parallel Sysplex.

sysplex couple data set. A couple data set that contains sysplex-wide data about systems, groups, and members that use XCF services. All MVS systems in a sysplex must have connectivity to the sysplex couple data set. See also couple data set.

Sysplex Timer. An IBM unit that synchronizes the time-of-day (TOD) clocks in multiple processors or processor sides. External Time Reference (ETR) is the MVS generic name for the IBM Sysplex Timer (9037).

system control element (SCE). Hardware that handles the transfer of data and control information associated with storage requests between the elements of the processor.

system console. In MVS, a console attached to the processor controller used to initialize an MVS system.

system log (SYSLOG). In MVS, the system log data set that includes all entries made by the WTL (write-to-log) macro as well as the hardcopy log. SYSLOG is maintained by JES in JES SPOOL space.

system management facilities (SMF). An optional control program feature of OS/390 and MVS that provides the means for gathering and recording information that can be used to evaluate system usage.

System Modification Program Extended (SMP/E). 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.

Systems Network Architecture (SNA). A description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of networks.

system trace. A chronological record of specific operating system events. The record is usually produced for debugging purposes.

T

temporary data set. A data set that is created and deleted in the same job.

terminal. A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a link.

terminal user. In systems with time-sharing, anyone who is eligible to log on.

tightly coupled. Multiple CPs that share storage and are controlled by a single copy of MVS. See also loosely coupled, tightly coupled multiprocessor.

tightly coupled multiprocessor. Any CPU with multiple CPs.

Time Sharing Option (TSO). An option on the operating system; for OS/390 the option provides interactive time sharing from remote terminals.

TOR. Terminal-owning region.

transaction. In APPC/MVS, a unit of work performed by one or more transaction programs, involving a specific set of input data and initiating a specific process or job.

transaction program (TP). For APPC/MVS, any program on MVS that issues APPC/MVS or CPI Communication calls, or is scheduled by the APPC/MVS transaction scheduler.

U

undelivered message. An action message or WTOR that cannot be queued for delivery to the expected console. MVS delivers these messages to any console with the UD console attribute in a system or sysplex.

uniprocessor (UP). A CPC that contains one CP and is not partitionable.

UP. Uniprocessor.

V

VM. Virtual Machine.

virtual telecommunications access method (VTAM). A set of programs that maintain control of the communication between terminals and application programs running under DOS/VS, OS/VS1, and OS/VS2 operating systems.

volume. (1) That portion of a single unit of storage which is accessible to a single read/write mechanism, for example, a drum, a disk pack, or part of a disk storage module. (2) A recording medium that is mounted and demounted as a unit, for example, a reel of magnetic tape, a disk pack, a data cell.

volume serial number. A number in a volume label that is assigned when a volume is prepared for use in the system.

volume table of contents (VTOC). A table on a direct access volume that describes each data set on the volume.

VSAM. Virtual Storage Access Method.

VTAM. Virtual Telecommunications Access Method.

VTOC. Volume table of contents.

W

wait state. Synonymous with waiting time.

waiting time. (1) The condition of a task that depends on one or more events in order to enter the ready condition. (2) The condition of a processing unit when all operations are suspended.

WLM. MVS workload management.

wrap mode. The console display mode that allows a separator line between old and new messages to move down a full screen as new messages are added. When the screen is filled and a new message is added, the separator line overlays the oldest message and the newest message appears immediately before the line.

write-to-log (WTL) message. A message sent to SYSLOG or the hardcopy log.

write-to-operator (WTO) message. A message sent to an operator console informing the operator of errors and system conditions that may need correcting.

write-to-operator-with-reply (WTOR) message. A message sent to an operator console informing the operator of errors and system conditions that may need correcting. The operator must enter a response.

WTL message. Write-to-log message

WTO message. Write-to-operator message

WTOR message. Write-to-operator-with-reply message.

X

XCF. Cross-system coupling facility.

XCF PR/SM policy. In a multisystem sysplex on PR/SM, the actions that XCF takes when one MVS system in the sysplex fails. This policy provides high availability for multisystem applications in the sysplex.

XCF-local mode. The state of a system in which XCF provides limited services on one system and does not provide signalling services between MVS systems. See also single-system sysplex.

XRF. Extended recovery facility.

IBM Redbooks evaluation

ABCs of OS/390 System Programming Volume 3
SG24-5653-00

Your feedback is very important to help us maintain the quality of IBM Redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
__Customer __Business Partner __Solution Developer __IBM employee
__None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

SG24-5653-00
Printed in the U.S.A.

SG24-5653-00