Interactive System Productivity Facility (ISPF)

Dialog Tag Language Guide and Reference

z/OS Version 1 Release 2.0

SC34-4824-01

Interactive System Productivity Facility (ISPF)

Dialog Tag Language Guide and Reference

z/OS Version 1 Release 2.0

SC34-4824-01

Note
Before using this document, read the general information under "Notices" on page 529.

Second Edition (October 2001)

This edition applies to ISPF for Version 1 Release 2 of the licensed program z/OS (program number 5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications by phone or fax. IBM Software Manufacturing Solutions takes publication orders between 8:30 a.m. and 7:00 p.m. eastern standard time (EST). The phone number is (800) 879-2755. The fax number is (800) 284-4721.

You can also order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for comments appears at the back of this publication. If the form has been removed, and you have ISPF-specific comments, address your comments to:

International Business Machines Corporation
Software Reengineering
Department G7IA / Building 503
Research Triangle Park, NC 27709-9990

FAX (United States & Canada): 1+800+227-5088
IBMLink (United States customers only): CIBMORCF@RALVM17
IBM Mail Exchange: USIB2HPD@VNET.IBM.COM
Internet: USIB2HPD@VNET.IBM.COM

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

Title and order number of this book
Page number or topic related to your comment

The ISPF development team maintains a site on the World-Wide Web. The URL for the site is:

http://www.software.ibm.com/ad/ispf

© Copyright International Business Machines Corporation 1989, 2001. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures ix

Preface xi
About This Book xi
Who Should Use This Book xi
Using LookAt to look up message explanations . . xi
Accessing licensed books on the Web xii

Summary of Changes xiii
| ISPF Product Changes xiii
| ISPF DM Component Changes xiii
| ISPF PDF Component Changes xv
| ISPF SCLM Component Changes xvi
ISPF Client/Server Component Changes xvi
ISPF User Interface Considerations xvi
ISPF Migration Considerations xvii
ISPF Profiles xvii
Year 2000 Support for ISPF xvii

What's in the z/OS V1R2.0 ISPF library? xix
z/OS V1R2.0 ISPF xix

The ISPF User Interface xxi
Some Terms You Should Know xxi
How to Navigate in ISPF without Using Action Bars xxii
How to Navigate in ISPF Using the Action Bar Interface xxii
Action Bars xxii
Action Bar Choices xxiv
Point-and-Shoot Text Fields xxvi
Function Keys xxvi
Selection Fields xxvii
Command Nesting xxviii

Elements and Features in z/OS . . . xxix

Part 1. Guide to the Dialog Tag Language (DTL) 1

Chapter 1. Introduction to the Dialog Tag Language (DTL) 3
Why the Dialog Tag Language (DTL)? 3
What Is the Dialog Tag Language? 4
Dialog Elements 5
Application Panels 5
Help Panels. 7
Messages 8
Application Command Table 9
Key Mapping Lists 9
Variables and Variable Classes 9
What Is the ISPF Conversion Utility? 9

Chapter 2. How to Use the Dialog Tag Language (DTL) 11
Syntax Conventions. 11
Attributes and Values 11
Tag Text 12
Text Formatting 13
Nesting tags 15
Markup Declarations 16
Declaring the Document Type 16
Including Comments in Your Markup 16
Defining Entities and Parameter Entities. . . . 18
Imbedding Source Files 23
Run-Time Substitution Variables 25
Predefined Entities 25

Chapter 3. Getting Started: Designing Application Panels 29
Defining Application Panels: The PANEL Tag . . . 29
The Panel Title 31
Panel Size (Width and Depth) 31
Key Mapping Lists 31
Associated Help Panels 32
Panel Defaults 32
Cursor Placement 32
Other Panel Attributes. 34
Defining Action Bars and Pull-Downs 36
Coding an Action Bar Definition 37
Pull-Down Choice Actions 38
Action Bar Help 39
Preselected Pull-Down Choices 40
Mnemonic Choice Selection 40
Pull-Down Choice Accelerator Support 41
Defining the Panel Body 42
Panel Instructions 42
The AREA Tag 43
Scrollable Areas 44
Multiple AREA Tags 47
The DYNAMIC AREA Tag 49
The GRAPHIC AREA Tag 50
The DIVIDER Tag 50
The REGION Tag 51
Defining a Command Area 55
Defining Panel Defaults 57

Chapter 4. Variables and Variable Classes 61
Declaring Variables 61
Defining Variable Classes 62
Variable Class Types 63
Character Variables 63
Numeric Variables 64
Variable Validation 65
Translate Lists 65
Validity Checks 70

© Copyright IBM Corp. 1989, 2001

Overriding Variable Classes 81

Chapter 5. Application Panel Fields . . 83
Field Prompts 83
Defining Data Fields 86
Data Field Width 88
Data Field Descriptions 88
Data Field Help 89
Other Data Field Attributes 90
Defining Selection Fields 93
Single-Choice Fields 94
Multiple-Choice Fields. 95
Menu-Choice Fields 97
Model-Choice Fields 98
Tutor-Choice Fields 99
Selection Field Help 99
Selection Width. 100
Other Selection Field Attributes 101
Data Columns 104
Defining List Fields 106
List Group Headings 108
List Column Width 110
Other List Column Attributes 110
Defining Group Headings 113
Defining Point-and-Shoot Fields 113

Chapter 6. Information Regions and Help Panels 115
Defining an Information Region 115
Defining Basic Text 116
Paragraphs 116
Headings 118
Lines 119
Examples. 120
Figures 121
Defining Lists 123
Note Lists 124
Simple Lists 124
Unordered Lists 126
Ordered Lists 128
Definition Lists 131
Parameter Lists. 135
Nesting Tags within Lists 136
Nesting Lists within Lists 138
Alerting Users: Notes, Warnings, Cautions, and Attention 139
Notes 139
Attention and Warning 142
Caution 143
Emphasizing Panel Text 144
Using Information Regions with Other Panel Elements 148
Help Panels 149

Chapter 7. Messages 159
Defining Messages 159
Specifying Message Severity 160
Short Messages. 161
Assigning Messages 161
Displaying Messages 163
Variable Substitution 164

Chapter 8. The Application Command Table 165
Defining the Application Command Table 165
Specifying Command Actions 167
Truncating Commands 168

Chapter 9. Defining Key Mapping Lists 171
Assigning Keys and Actions 171
ISPF Default Key List. 172
Displaying Keys 172
Defining Help for Key List 173

Chapter 10. Using the Conversion Utility 175
Using the ISPF-Supplied Invocation Panels . . . 175
Invocation Panel 175
Panel Input Fields 178
Panel Options 179
Converting Multiple DTL Source Files 182
Calling Help 182
Using CUA Panel Attributes 182
Conversion Utility Syntax 183
Conversion Utility General Information 189
ISPF Conversion Utility Messages 193
Return Codes 193
Conversion Results 193
Conversion Utility File Names. 194

Part 2. Dialog Tag Language (DTL) Reference 197

Chapter 11. How to Read the Syntax Diagrams 199

Chapter 12. Markup Declarations and DTL Macro Reference 201
Document-Type Declaration 201
Description 201
Example 201
Entity Declarations 202
Description 203
Conditions 204
Example 204
Sample Entity Definitions 205
DTL Macros 207

Chapter 13. Tag Reference 211
Rules for Variable Names 211
Rules for "%variable" Names 211
AB (Action Bar) 211
Description 212
Conditions 212
Nested Tags 213
Example 213
ABC (Action Bar Choice) 214
Description 215

Conditions 215
Nested Tags 215
Example 215
ACTION (Action) 217
Description 221
Conditions 222
Nested Tags 222
Example 222
AREA (Area) 223
Description 225
Conditions 226
Nested Tags 226
Example 227
ASSIGNI (Assignment List Item) 229
Description 230
Conditions 230
Nested Tags 230
Example 230
ASSIGNL (Assignment List) 231
Description 231
Conditions 231
Nested Tags 231
Example 231
ATTENTION (Attention) 232
Description 232
Conditions 233
Nested Tags 233
Example 233
ATTR (Attribute) 234
Description 237
Conditions 237
Nested Tags 237
Example 237
BOTINST (Bottom Instruction). 238
Description 238
Conditions 238
Nested Tags 239
Example 239
CAUTION (Caution) 240
Description 240
Conditions 240
Nested Tags 240
Example 241
CHDIV (Choice Divider) 242
Description 242
Conditions 242
Nested Tags 242
Example 242
CHECKI (Validity Check Item) 244
Description 252
Conditions 252
Nested Tags 252
Example 252
CHECKL (Validity Check List). 253
Description 253
Conditions 253
Nested Tags 253
Example 253
CHOFLD (Choice Data Field) 254
Description 259
Conditions 259
Nested Tags 259
Example 260
CHOICE (Selection Choice). 261
Description 266
Conditions 267
Nested Tags 267
Example 267
CMD (Command Definition) 268
Description 269
Conditions 269
Nested Tags 270
Example 270
CMDACT (Command Action) 270
Description 272
Conditions 272
Nested Tags 272
Example 272
CMDAREA (Command Area) 273
Description 278
Conditions 278
Nested Tags 278
Example 279
CMDTBL (Command Table) 280
Description 280
Conditions 281
Nested Tags 281
Example 281
COMMENT (Comment) 282
Description 282
Conditions: 283
Nested Tags 283
Example 283
COMPOPT (Compiler Options) 284
Description 285
Conditions: 285
Nested Tags 285
Example 285
COPYR (Copyright) 286
Description 286
Conditions 286
Nested Tags 286
Example 286
DA (Dynamic Area) 287
Description 291
Conditions 291
Nested Tags 292
Example 292
DD (Definition Description) 292
Description 292
Conditions 292
Nested Tags 293
Example 293
DDHD (Definition Description Header) 294
Description 294
Conditions 294
Nested Tags 294
Example 295
DIVIDER (Area Divider) 296
Description 297
Conditions 297
Nested Tags 297

Example 297
DL (Definition List) 298
Description 299
Conditions 300
Nested Tags 300
Example 300
DLDIV (Definition List Divider) 302
Description 303
Conditions 303
Nested Tags 303
Example 303
DT (Definition Term) 304
Description 305
Conditions 305
Nested Tags 305
Example 305
DTACOL (Data Column) 306
Description 310
Conditions 310
Nested Tags 311
Example 311
DTAFLD (Data Field). 312
Description 321
Conditions 321
Nested Tags 321
Example 321
DTAFLDD (Data Field Description) 323
Description 323
Conditions 323
Nested Tags 323
Example 323
DTDIV (Definition Term Divider). 324
Description 324
Conditions 324
Nested Tags 324
Example 324
DTHD (Definition Term Header) 326
Description 326
Conditions 326
Nested Tags 326
Example 327
DTHDIV (Definition Term Header Divider) . . . 328
Description 328
Conditions 328
Nested Tags 328
Example 328
FIG (Figure) 329
Description 330
Conditions 330
Nested Tags 330
Example 331
FIGCAP (Figure Caption) 331
Description 332
Conditions 332
Nested Tags 332
Example 332
GA (Graphic Area) 333
Description 334
Conditions 335
Nested Tags 335
Example 335
GENERATE (Generate) 335
Description 336
Conditions 336
Nested Tags 336
Example 336
GRPHDR (Group Header) 337
Description 338
Conditions 339
Nested Tags 339
Example 339
HELP (Help Panel) 340
Description 345
Conditions 346
Nested Tags 346
Example 347
HELPDEF (Help default) 349
Description 351
Conditions 351
Nested Tags 352
Example 352
Hn (Heading) 352
Description 352
Conditions 353
Nested Tags 353
Example 353
HP (Highlighted Phrase) 354
Description 355
Conditions 355
Nested Tags 355
Example 355
INFO (Information Region) 356
Description 357
Conditions 357
Nested Tags 357
Example 358
KEYI (Key Item) 358
Description 360
Conditions 360
Nested Tags 360
Example 360
KEYL (Key List) 361
Description 362
Conditions 363
Nested Tags 363
Example 363
LI (List Item) 364
Description 364
Conditions 365
Nested Tags 365
Example 366
LINES (Lines) 366
Description 367
Conditions 367
Nested Tags 367
Example 367
LIT (Literal) 368
Description 369
Conditions 369
Nested Tags 369
Example 369
LP (List Part) 370

Description 370
Conditions 370
Nested Tags 370
Example 371
LSTCOL (List Column) 372
Description 378
Conditions 379
Nested Tags 379
Examples 379
LSTFLD (List Field) 382
Description 384
Conditions 384
Nested Tags 385
Example 385
LSTGRP (List Group) 386
Description 387
Conditions 387
Nested Tags 388
Example 388
LSTVAR (List Variable) 389
Description 390
Conditions 390
Nested Tags 390
Example 390
M (Mnemonic) 392
Description 392
Conditions 392
Nested Tags 393
Example 393
MSG (Message) 393
Description 396
Conditions 396
Nested Tags 396
Example 397
MSGMBR (Message Member) 397
Description 398
Conditions 398
Nested Tags 399
Example 399
NOTE (Note) 400
Description 400
Conditions 401
Nested Tags 401
Example 401
NOTEL (Note List) 402
Description 402
Conditions 402
Nested Tags 403
Example 403
NT (Note) 404
Description 404
Conditions 404
Nested Tags 404
Example 405
OL (Ordered List) 405
Description 406
Conditions 407
Nested Tags 407
Example 407
P (Paragraph) 408
Description 408
Conditions 409
Nested Tags 409
Example 409
PANDEF (Panel Default) 410
Description 413
Conditions 413
Nested Tags 414
Example 414
PANEL (Panel) 415
Description 423
Conditions 423
Nested Tags 423
Example 424
PARML (Parameter List) 425
Description 426
Conditions 427
Nested Tags 427
Example 427
PD (Parameter Description) 428
Description 428
Conditions 428
Nested Tags 428
Example 429
PDC (Pull-Down Choice) 430
Description 432
Conditions 432
Nested Tags 432
Example 432
PDSEP (Pull-Down Separator) 434
Description 434
Conditions 434
Nested Tags 434
Example 434
| PLDIV (Parameter List Divider) 435
| Description 436
| Conditions 436
| Nested Tags 436
| Example 436
PNLINST (Panel Instruction) 437
Description 438
Conditions 438
Nested Tags 438
Example 438
PS (Point-and-Shoot) 439
Description 441
Conditions 441
Nested tags 441
Example 441
PT (Parameter Term) 442
Description 443
Conditions 443
Nested Tags 443
Example 443
| PTDIV (Parameter Term Divider) 444
| Description 444
| Conditions 444
| Nested Tags 444
| Example 444
REGION (Region) 445
Description 448
Conditions 448

Nested Tags 448
Help Panel 449
Example 449
RP (Reference Phrase) 452
Description 453
Conditions 453
Nested Tags 453
Example 454
SELFLD (Selection Field) 455
Description 465
Conditions 466
Nested Tags 467
Example 467
SL (Simple List) 470
Description 470
Conditions 470
Nested Tags 471
Example 471
SOURCE (Source) 472
Description 472
Conditions 473
Nested Tags 473
Example 473
T (Truncation) 474
Description 474
Conditions 474
Nested Tags 474
Example 474
TEXTLINE (Text Line) 475
Description 475
Conditions 475
Nested Tags 475
Example 475
TEXTSEG (Text Segment) 475
Description 476
Conditions 476
Nested Tags 476
Example 476
TOPINST (Top Instruction) 477
Description 478
Conditions 478
Nested Tags 478
Example 478
UL (Unordered List) 479
Description 480
Conditions 480
Nested Tags 481
Example 481
VARCLASS (Variable Class) 482
Description 484
Conditions 485
Nested Tags 485
Example 485
VARDCL (Variable Declaration) 486
Description 486
Conditions 486
Nested Tags 486
Example 486
VARLIST (Variable List) 487
Description 487
Conditions 487
Nested Tags 488
Example 488
VARSUB (Variable Substitution) 489
Description 489
Conditions 489
Nested Tags 489
Example 490
WARNING (Warning) 491
Description 491
Conditions 491
Nested Tags 491
Example 492
XLATI (Translate Item) 493
Description 494
Conditions 494
Nested Tags 494
Example 494
XLATL (Translate List) 495
Description 496
Conditions 496
Nested Tags 496
Example 496
XMP (Example). 497
Description 497
Conditions 498
Nested Tags 498
Example 498

Part 3. Appendixes 501

Appendix. Dialog Tag Language (DTL) Tags 503
Tag Summary 503

Notices 529
Programming Interface Information 530
Trademarks 530

Index 533

Figures

1. Panel with an Action Bar Pull-Down Menu xxiii
2. Pop-Up Selected from an Action Bar Pull-Down xxiv
3. Panel with an Action Bar and Point-and-Shoot Fields xxiv
4. An Unavailable Choice on a Pull-Down xxv
5. Application Panel 6
6. Help Panel 7
7. Message Displayed in Message Area 8
8. Text Formatting Rules 14
9. Entity Reference for Text Substitution 19
10. Entity Reference for Text Substitution 20
11. Entity Reference for Text Substitution and File Embedding 21
12. Parameter Entities 23
13. Cursor Placement 33
14. Action Bar and Pull-Down 37
15. Preselected Pull-Down Choice 41
16. Top and Bottom Instructions 43
17. AREA MARGINW=10 44
18. Scrollable Panel Area 46
19. Application Panel Area. 46
20. Scrollable Panel Area 47
21. Multiple Horizontal Areas 49
22. Area Dividers 51
23. Vertical Region 52
24. Horizontal Region 53
25. Horizontal Region 55
26. Command Area 56
27. Command Area 57
28. Variable Translation Results 66
29. Variable Translation 67
30. Variable Translation 68
31. Variable Translation 69
32. Variable Translation 70
33. Prompt Locations 84
34. Prompt Widths 85
35. Data Fields. 87
36. Data Field Description 89
37. Single-Choice Selection Field 95
38. Multiple-Choice Selection Field 97
39. Sample Option Menu 98
40. Selection Field SELWIDTH Attribute 101
41. Data Column 106
42. List Field 108
43. List Group 109
44. Paragraph. 117
45. Multiple Paragraphs 118
46. Headings (H1-H2) 119
47. LINES 120
48. XMP 121
49. Figure with Rules 122
50. Figure Caption 123
51. Simple List 125
52. Compact Simple List 126
53. Unordered List 127
54. Nested Unordered Lists 128
55. Ordered List 129
56. Nested Ordered Lists 131
57. Definition List 132
58. Definition List (BREAK=ALL) 134
59. Definition List (BREAK=FIT) 135
60. Parameter List 136
61. Nested Paragraph within a List 137
62. List Part 138
63. Nested Unordered List in a Definition List 138
64. Note (NOTE tag) 140
65. Notel (NOTEL tag). 141
66. Note (NT tag) 142
67. Warning 143
68. Caution 144
69. Highlighted Phrase Example 146
70. Reference Phrase Example 147
71. Reference Phrase Example of Help Attribute 148
72. Information Region 149
73. Help Panel 152
74. Help Panel (Example 1 of 4) 153
75. Help Panel (Example 2 of 4) 154
76. Help Panel (Example 3 of 4) 154
77. Help Panel (Example 4 of 4) 155
78. Help Panel (Example 1 of 3) 156
79. Help Panel (Example 2 of 3) 156
80. Help Panel (Example 3 of 3) 157
81. Data Field and Message 163
82. Displayed Function Key Area 174
83. Conversion Utility Invocation Panel (ISPCP01) 175
84. Conversion Utility Invocation Panel (Middle) 176
85. Conversion Utility Invocation Panel 176
86. Conversion Utility Invocation Panel 177
87. Conversion Utility Invocation Panel (End) 177
88. Panel ISPCP04 182
89. ISPF Dialog Tag Language Conversion Utility - Confirm Cancel 191
90. ISPF Dialog Tag Language Conversion Utility - Recursive invoke 192
91. Entity Declarations. 202
92. Entities and Parameter Entities. 205
93. Action Bar 214
94. Action Bar Choices. 216
95. Application Panel Area 228
96. Application Panel Area 228
97. Application Panel Area 229
98. Attention Statement 234
99. Bottom Instructions 239
100. Caution Statement 241
101. Choice Divider 244
102. Choice Data Fields. 261
103. Selection Field Choices 268
104. Command Area 279
105. Comment text added to a panel 284
106. Copyright statement added to a panel 287

© Copyright IBM Corp. 1989, 2001

107. Definition Descriptions 294
108. Definition Description Header 295
109. Area Dividers 298
110. Definition List (BREAK=NONE) 301
111. Definition List (BREAK=FIT) 301
112. Definition List (BREAK=ALL) 302
113. Definition List Dividers 304
114. Definition Terms 306
115. 310
116. Data Column 312
117. Data Fields 322
118. Data Field Descriptions 324
119. Definition Term Divider 326
120. Definition Term Header 327
121. Definition Term Header Divider 329
122. Figure 331
123. Figure Caption 333
124. Generated Panel 337
125. Group Heading 340
126. Help Panel (Example 1 of 3) 348
127. Help Panel (Example 2 of 3) 348
128. Help Panel (Example 3 of 3) 349
129. Headings 354
130. HP (Highlighted Phrase) 356
131. Information Region 358
132. Key Items 361
133. Function Keys 364
134. List Items 366
135. LINES 368
136. List Part 372
137. List Columns 380
138. List Columns 382
139. List Field 386
140. List Group 389
141. List Variable 391
142. Messages 397
143. Message Member 400
144. Note 401
145. NOTEL 403
146. NT 405
147. Ordered Lists 408
148. Paragraphs 410
149. Application Panel 425
150. Parameter List 428
151. Parameter Descriptions 430
152. Pull-Down Choices 433
153. Pull-Down Separator 435
154. Parameter List Divider 437
155. Panel Instructions 439
156. Point-and-Shoot Fields 442
157. Parameter Terms 444
158. Parameter Term Divider 445
159. Regions 450
160. Using WIDTH and DEPTH Attributes 452
161. Reference Phrase Example 454
162. Reference Phrase Example 454
163. Selection Fields 468
164. Selection Menu 469
165. Simple List 471
166. Top Instructions. 479
167. Unordered List 482
168. Variable Substitution 490
169. Warning Statement. 493
170. Example 499

Preface

This book describes how to use the Dialog Tag Language (DTL), the tag-based markup language you use to create the following ISPF dialog elements:

- Application panels
- Help panels
- Application command tables
- Messages
- Key lists.

It also explains how to use the ISPF conversion utility to convert the source files that contain DTL markup into ISPF panel language source or execution time format.

About This Book

This book is organized into two sections:

- Part 1 guides you through the steps involved in using the Dialog Tag Language for designing, defining, and converting dialog elements for ISPF applications.
- Part 2 provides an alphabetical reference of the Dialog Tag Language tags and markup declarations.

Who Should Use This Book

This book is for ISPF application developers who want to use the Dialog Tag Language (DTL) to create dialog elements for ISPF applications.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages. You can also use LookAt to look up explanations of system abends.

Using LookAt to find information is faster than a conventional search because LookAt goes directly to the explanation.

LookAt can be accessed from the Internet or from a TSO command line.

You can use LookAt on the Internet at:

```
www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html
```

To use LookAt as a TSO command, LookAt must be installed on your host system. You can obtain the LookAt code for TSO from the LookAt Web site by clicking on the **News and Help** link or from the *z/OS Collection*, SK3T-4269.

To find a message explanation from a TSO command line, simply enter: **lookat** *message-id* as in the following:

```
lookat iec192i
```

This results in direct access to the message explanation for message IEC192I.

To find a message explanation from the LookAt Web site, simply enter the message ID and select the release you are working with.

© Copyright IBM Corp. 1989, 2001

Note: Some messages have information in more than one book. For such messages, LookAt prompts you to choose which book to open.

Accessing licensed books on the Web

z/OS Version 1 Release 2.0 licensed documentation in PDF format is available on the Internet at the IBM Resource Link Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed books are available only to customers with a z/OS Version 1 Release 2.0 license. Access to these books requires an IBM Resource Link Web userid and password, and a key code. With your z/OS Version 1 Release 2.0 order you received a memo that includes this key code.

To obtain your IBM Resource Link Web userid and password log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **User Profiles** located on the left-hand navigation bar.
3. Click on **Access Profile.**
4. Click on **Request Access to Licensed books.**
5. Supply your key code where requested and click on the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed. After your request is processed you will receive an e-mail confirmation.

Note: You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

To access the licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **Library**.
3. Click on **zSeries**.
4. Click on **Software**.
5. Click on **z/OS Version 1 Release 2.0**.
6. Access the licensed book by selecting the appropriate element.

Summary of Changes

z/OS V1R2.0 ISPF contains the following changes and enhancements:
- ISPF Product and Library Changes
- ISPF Dialog Manager Component Changes (including DTL changes)
- ISPF PDF Component Changes
- ISPF SCLM Component Changes
- ISPF Client/Server Component Changes

ISPF Product Changes

Changes to the ZENVIR variable. Characters 1 through 8 contain the product name and sequence number in the format *ISPF x.y*, where x.y indicates:
- <= 4.2 means the version.release of ISPF
- = 4.3 means ISPF for OS/390 release 2
- = 4.4 means ISPF 4.2.1 and ISPF for OS/390 release 3
- = 4.5 means ISPF for OS/390 Version 2 Release 5.0
- = 4.8 means ISPF for OS/390 Version 2 Release 8.0
- = 5.0 means ISPF for OS/390 Version 2 Release 10.0
- OR
- = 5.0 means ISPF for z/OS Version 1 Release 1.0
- = 5.2 means ISPF for z/OS Version 1 Release 2.0

The ZENVIR variable is used by IBM personnel for internal purposes. The x.y numbers DO NOT directly correlate to an ISPF release number in all cases. For example, as shown above, a ZENVIR value of 4.3 DOES NOT mean ISPF Version 4 Release 3. NO stand-alone version of ISPF exists above ISPF Version 4 Release 2 Modification 1.

The ZOS390RL variable contains the ISPF release on your system.

The ZISPFOS system variable contains the level of ISPF code that is running as part of the operating system release on your system. This might or might not match ZOS390RL. For this release, the variable contains **ISPF for z/OS 01.02.00**.

New system variables:

ZDAYOFWK
The day of the week.

The ISRDDN utility is now documented in the ISPF User's Guide.

ISPF DM Component Changes

The DM component of ISPF includes the following new functions and enhancements:
- Add support for "VER(&variable,IPADDR4)".
- Add the NOSETMSG parameter to the CONTROL Service.
- Add the LFORMAT parameter to the VDEFINE Service to allow defining like format variables in a list.
- Change tutorial processing to eliminate the "End of data" message on scrollable area panels that display the entire scrollable area on the screen (no More: + - is displayed). This change eliminates the extra enter the user had to execute before continuing to the next panel.

© Copyright IBM Corp. 1989, 2001

- Issue a TSO line message when a help panel is not found and continue the dialog. Previously ISPF issued a severe error message when a help panel could not be found.
- Display a message indicating a message is not found when running in Dialog Test and allow the dialog to continue.
- Add support for extended SBCS and DBCS CCSIDs:
  - 1159 Traditional Chinese
  - 1364 Korean
  - 1371 Traditional Chinese
  - 1388 Simplified Chinese
  - 1390 Japanese
  - 1399 Japanese
- Add new Z variables to support 5 character code pages and character sets, ZTERMCP5 and ZTERMCS5 respectively.
- Add new variable ZDAYOFWK to show the day of the week.
- Enhance the Reflist function of TEST option 7.6 to allow better list management.
- Enhance Locate and Find for Dialog Test Variables (option 7.3).
- A new exec called ISPCMDTB to convert ISPF command tables to DTL.
- A new Configuration Table variable to allow SCROLL defaults.
- A new Configuration Table variable to allow STATUS AREA defaults.

ISPDTLC enhancements:

ISPDTLC changes include new invocation options, new tags, and new tag. attributes as ISPF extensions to the Dialog Tag Language

General improvements:

- New invocation options:
  - no new invocation options in this release
- New tags:
  - DLDIV, DTDIV, DTHDIV for dividers within the DL tag
  - PLDIV, PTDIV for dividers within the PARML tag
- Replication added to predefined entities. For example, &GTSYM(5); will create the string '>>>>>' in the substituted text.
- National language text strings are now accessible as entities. For example, &command; will create the string 'Command' or its translated equivalent in the substituted text.
- New ENTITY keywords COPIES, X2C and ATTR.
- New macro tag default initialization processing syntax.

  ```
  <?dummy ?var=value>
  ```
- New Predefined ENTITY keywords cmdpmt (&cmdpmt;) and rptr (&rptr;).

New or changed tag attributes:

Tag name	Attribute update
CHECKI	Add support for "VER(&variable, IPADDR4)"
COMPOPT	Add ADD.

Tag name	Attribute update
DL	Add FORMAT. Support multiple DT tags for each DD tag. Change TSIZE to support multiple values. Each TSIZE value implies a DT tag.
DT	Add FORMAT, NOSKIP.
DTAFLD	Add AUTOTYPE, AUTOVOL, AUTODMEM.
HELP	Add ZUP, ZCONT.
Hn	Add COMPACT.
HP	Add INTENSE.
NOTE	Add NOSKIP.
NT	Add NOSKIP.
PANEL	Add ZUP, ZCONT, AUTONRET, AUTOTCMD.
PARML	Add FORMAT. Support multiple PT tags for each PD tag. Change TSIZE to support multiple values. Each TSIZE value implies a PT tag.
PT	Add FORMAT, SKIP.
SELFLD	Add SELCHECK.
	Support INIT=init-value for single-choice selection fields.

ISPF PDF Component Changes

The ISPF PDF component contains the following new functions and enhancements:

- A MEMBER command has been added to data set list (option 3.4) to allow the partitioned data sets in the list to be searched for a specific member.
- When the EDIT service is specified with an initial macro, parameters can now be specified for the initial macro.
- A FIND command has been added to member list to allow a string to be searched for in any of the displayed statistics.
- A SRCHFOR command has been added to data set list to allow SuperC to be invoked to search the listed data sets for strings.
- Move/Copy will now dynamically calculate the sized for the IEBCOPY SYSUT3 and SYSUT4 data sets.
- A QUERYENQ service has been added to retrieve ENQ information about a data set in use.
- LMF has been removed from the ISPF product.
- A new SuperC option FINDALL has been added to specify that all strings must be found to issue a ″strings found″ return code.
- LMPRINT will now allow the INDEX parameter to be specified for a record format U data set.
- Foreground and Batch now support the z/OS C/C++ compiler.

- A new AUTOTYPE command can be set to a PFKEY to retrieve a data set name or pattern entered on a panel based on data sets that start with that partial name.
- Data sets with an LRECL less than 10 bytes can be edited or viewed.
- The Edit CUT and PASTE command defaults have been added to the ISPF Configuration Table.
- The Edit CUT and PASTE default behaviors have been modified to use CUT REPLACE and PASTE KEEP.
- The BARRIER keyword has been added to the SELECT for Edit macros.
- A program called ISREMSPY that can be invoked from an Edit macro to display the current Edit data.
- The Edit macro commands CURSOR, LINENUM and DISPLAY_LINES can retrieve line numbers greater than 999999.

ISPF SCLM Component Changes

The ISPF SCLM component contains the following new functions and enhancements:

- Several enhancements to the Library Utility:
  - A member action to initiate Promotion on a member.
  - REFRESH command to update the member list contents.
  - HIER ON|OFF command to switch between full hierarchy view and single group view.
  - Edit action can create a new member when entered on the command line.
  - Ability to select deletion of accounting data or build map only.
- New FLMLRBLD macro to select automated rebuild for members with a specified language on promotion to listed groups.
- Improved edit models for SCLM services.
- VOL keyword on the FLMCPYLB and FLMSYSLB macros allowing reference to uncatalogued data sets.
- VIO keyword on the FLMALLOC macro to override the SCLM-calculated default unit of DASD or VIO for temporary data sets.
- Supplied parsers and translators are all loaded RMODE(31).

ISPF Client/Server Component Changes

The ISPF Client/Server Component enables a panel to be displayed unchanged (except for panels with graphic areas) at a workstation using the native display function of the operating system of the workstation. ISPF manuals call this "running in GUI mode."

There are no changes to the ISPF Client/Server for this release.

ISPF User Interface Considerations

Many changes have been made to the ISPF Version 4 user interface to conform to CUA guidelines. If you prefer to change the interface to look and act more like the Version 3 interface, you can do the following:

- Use the CUAATR command to change the screen colors
- Use the ISPF Settings panel to specify that the TAB or HOME keys position the cursor to the command line rather than to the first action bar item

- Set the command line to the top of the screen by deselecting ***Command line at bottom*** on the ISPF Settings panel
- Set the primary keys to F13–24 by selecting 2 for Primary range on the Tailor Function Key Definition Display panel
- Use the KEYLIST OFF command to turn keylists off
- Use the PSCOLOR command to change point-and-shoot fields to blue.
- Change the DFLTCOLR field in the PDF configuration table ISRCONFG to disable action bars and or edit highlighting

ISPF Migration Considerations

When migrating to OS/390 V2R8.0 or higher for the first time, you must convert your ISPF customization to the new format. Refer to the section entitled ***The ISPF Configuration Table*** in the *ISPF Planning and Customizing manual.*

When migrating from one version of ISPF to another, you must be sure to reassemble and re-link the SCLM project definition.

Note: If you are migrating to z/OS V1R2.0 from OS/390 V2R10.0, there are no migration actions necessary. If you are migrating to z/OS V1R2.0 from a prior release of OS/390, follow the migration actions for OS/390 V2R10.0.

ISPF Profiles

Major changes were made to the ISPF profiles for ISPF Version 4.2 and OS/390 Version 1 Release 1.0 ISPF. The profiles for ISPF Version 3 and the profiles for OS/390 ISPF are not compatible. If you are moving back and forth between an ISPF Version 3 system and OS/390 V1R1.0 or higher system, you must run with separate profiles. Profiles for OS/390 V1R1.0 and higher are compatible with each other.

Year 2000 Support for ISPF

ISPF is fully capable of using dates for the year 2000 and beyond. All of your existing applications should continue to run (some may need minor changes, as explained below) when the year 2000 comes. The base support for the year 2000 was added to OS/390 Version 1 Release 2.0, but the same level of support is available for ISPF Version 3.5, ISPF Version 4, and OS/390 Version 1 Release 1.0 as well. To get support for the earlier versions, be sure that your system has the correct APARs installed. All ISPF APARs that add or correct function relating to the year 2000 contain the YR2000 identifier in the APAR text. You should search for these APARs to ensure you have all the function available.

What function is included?

- ISPF Dialog variable ZSTDYEAR now correctly shows the year for dates past 1999. Earlier versions always showed the first 2 characters of the year as 19.
- A new ISPF dialog variable (ZJ4DATE) is available for Julian dates with a 4–digit year.
- An ISPF Configuration Table field enables PDF to interpret 2 character year dates as either a 19xx or 20xx date. The default value is 65. Any 2-character year date whose year is less than or equal to this value is considered a 20xx date, anything greater than this value is considered 19xx. To see what value has been set by the ISPF Configuration Table, use the new ZSWIND variable.
- New parameters in the LMMSTATS service (CREATED4 and MODDATE4) for specifying 4-character year dates. All existing parameters still exist and you can

continue to use them. If both the 2-character year date parameters (CREATED and MODDATE) and the 4-character year date parameters (CREATED4 and MODDATE4) are specified, the 2-character versions are used.

- Dialog variables ZLC4DATE and ZLM4DATE have been added.
  - You ***can*** set them before making an LMMREP or LMMADD call. Do this to specify a 4-character *created* or *last modified* date to set in the ISPF statistics.
  - They ***are*** set by LMMFIND, LMMLIST and LMMDISP to the current value of the created and last modified dates in the ISPF statistics.

What might need to change? Some minor changes to your existing ISPF dialogs might be necessary, especially in ISPF dialogs that use the Library Access Services to manipulate ISPF member statistics.

- For those services that accept both 4-character year dates and 2-character year dates, you can specify one or the other. If you specify both, the 2-character year date is used to avoid affecting existing dialogs. When the 2-character year date is used, the configuration table field mentioned above is used to determine whether the date should be interpreted as 19xx or 20xx.
- ISPF will not necessarily show 4-character dates in all circumstances but it will process them correctly. For example, a member list might only display 2-character year dates but will sort those dates in the proper order.
- SCLM stores dates past the year 1999 in a new internal format. If an accounting file contains dates in this new format, it cannot be processed by a system without year 2000 support. Accounting files without dates past 1999 can be processed with or without the year 2000 support.
- LMF has been removed from the ISPF product. For information about how to convert from LMF to SCLM refer to the ***ISPF Planning and Customizing*** manual.

What's in the z/OS V1R2.0 ISPF library?

You can order the ISPF books using the numbers provided below.

z/OS V1R2.0 ISPF

Title	Order Number
z/OS V1R2.0 ISPF Dialog Tag Language Guide and Reference	SC34-4824-01
z/OS V1R2.0 ISPF Planning and Customizing	GC34-4814-01
z/OS V1R2.0 ISPF User's Guide Volume I	SC34-4822-01
z/OS V1R2.0 ISPF User's Guide Volume II	SC34-4823-01
z/OS V1R2.0 ISPF Services Guide	SC34-4819-01
z/OS V1R2.0 ISPF Dialog Developer's Guide and Reference	SC34-4821-01
z/OS V1R2.0 ISPF Reference Summary	SC34-4816-01
z/OS V1R2.0 ISPF Edit and Edit Macros	SC34-4820-01
z/OS V1R1.0 ISPF Library Management Facility	SC34-4825-01
z/OS V1R2.0 ISPF Messages and Codes	SC34-4815-01
z/OS V1R2.0 ISPF Software Configuration and Library Manager Project Manager's and Developer's Guide	SC34-4817–01
z/OS V1R2.0 ISPF Software Configuration and Library Manager Reference	SC34-4818-01
Entire library Bill of Forms	SBOF-8570

© Copyright IBM Corp. 1989, 2001

The ISPF User Interface

ISPF provides an action bar-driven interface that exploits many of the usability features of Common User Access (CUA) interfaces. Refer to *Object-Oriented Interface Design: IBM Common User Access Guidelines* for additional information.

The panels look different than in Version 3: all screens are in mixed case, and most have action bars at the top. These action bars give you a new way to move around in the product as well as access to command nesting. Command nesting allows you to *suspend* an activity while you perform a new one rather than having to end a function to perform another function.

This chapter primarily explains the action bar-driven interface and the use of ISPF's graphical user interface (GUI).

Some Terms You Should Know

The following terms are used in this book:

action bar. The area at the top of an ISPF panel that contains choices that give you access to actions available on that panel. When you select an action bar choice, ISPF displays a *pull-down menu*.

pull-down menu. A list of numbered choices extending from the selection you made on the action bar. The action bar selection is highlighted; for example, Utilities in Figure 1 on page xxiii appears highlighted on your screen. You can select an action either by typing in its number and pressing Enter or by selecting the action with your cursor. ISPF displays the requested panel. If your choice contains an *ellipsis* (...), ISPF displays a *pop-up window*. When you exit this panel or pop-up, ISPF closes the pull-down and returns you to the panel from which you made the initial action bar selection.

ellipsis. Three dots that follow a pull-down choice. When you select a choice that contains an ellipsis, ISPF displays a *pop-up* window.

pop-up window. A bordered temporary window that displays over another panel.

modal pop-up window. A type of window that requires you to interact with the panel in the pop-up before continuing. This includes cancelling the window or supplying information requested.

modeless pop-up window. A type of window that allows you to interact with the dialog that produced the pop-up before interacting with the pop-up itself.

point-and-shoot text. Text on a screen that is cursor-sensitive. See "Point-and-Shoot Text Fields" on page xxvi for more information.

push button. A rectangle with text inside. Push buttons are used in windows for actions that occur immediately when the push button is selected (available only when you are running in GUI mode).

function key. In previous releases of ISPF, a programmed function (PF) key. *This is a change in terminology only*.

select. In conjunction with point-and-shoot text fields and action bar choices, this means moving the cursor to a field and simulating Enter.

mnemonics. Action bar choices can be defined with a underscored letter in the action bar choice text. In host mode you can access the action bar choice with the ACTIONS command and parameter 'x', where 'x' is the underscored letter in the action bar choice text. In GUI mode you can use a *hot key* to access a choice on the action bar; that is, you can press the ALT key in combination with the letter that is underscored in the action bar choice text.

© Copyright IBM Corp. 1989, 2001

How to Navigate in ISPF without Using Action Bars

If you use a non-programmable terminal to access z/OS V1R2.0 ISPF and you do not want to take advantage of the command nesting function, you can make selections the same way you always have: by typing in a selection number and pressing Enter.

How to Navigate in ISPF Using the Action Bar Interface

Most ISPF panels have action bars at the top; the choices appear on the screen in white by default. Many panels also have point-and-shoot text fields, which appear in turquoise by default. The panel shown in Figure 3 on page xxiv has both.

Action Bars

Action bars give you another way to move through ISPF. If the cursor is located somewhere on the panel, there are several ways to move it to the action bar:

- Use the cursor movement keys to manually place the cursor on an action bar choice.
- Type ACTIONS on the command line and press Enter to move the cursor to the first action bar choice.
- Press F10 (Actions) or the Home key to move the cursor to the first action bar choice.

  If mnemonics are defined for action bar choices, you can:

  – In 3270 mode, on the command line, type ACTIONS and the mnemonic letter that corresponds to an underscored letter in the action bar choice text. This results in the display of the pull-down menu for that action bar choice.

  – In 3270 mode, on the command line enter the mnemonic letter that corresponds to an underscored letter in the action bar choice text, and press the function key assigned to the ACTIONS command. This results in the display of the pull-down menu for that action bar choice.

  – In GUI mode, you can use a *hot key* to access a choice on an action bar or on a pull-down menu; that is, you can press the ALT key in combination with the mnemonic letter that is underscored in the choice text to activate the text.

Use the tab key to move the cursor among the action bar choices. If you are running in GUI mode, use the right and left cursor keys.

Notes:

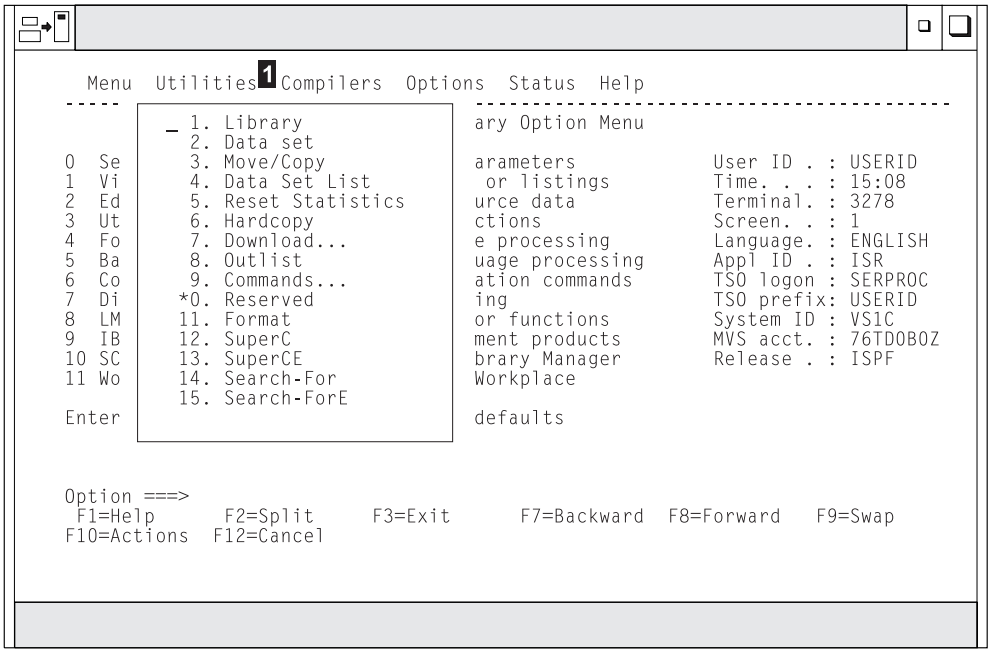
1. ISPF does not provide a mouse emulator program. This book uses *select* in conjunction with point-and-shoot text fields and action bar choices to mean moving the cursor to a field and simulating Enter.

   Note: Some users program their mouse emulators as follows:

   - Mouse button 1 – to position the cursor to the pointer and simulate Enter
   - Mouse button 2 – to simulate F12 (Cancel).

2. If you want the Home key to position the cursor at the first input field on an ISPF panel, type `SETTINGS` on any command line and press Enter to display the ISPF Settings panel. Deselect the **Tab to action bar choices** option.
3. If you are running in GUI mode, the Home key takes you to the beginning of the current field.

When you select one of the choices on the action bar, ISPF displays a pull-down menu. Figure 1 shows the pull-down menu displayed when you select Utilities on the ISPF Primary Option Menu action bar.



1 The selected action bar choice is highlighted.

Figure 1. Panel with an Action Bar Pull-Down Menu

To select a choice from the Utilities pull-down menu, type its number in the entry field (underlined) and press Enter or select the choice. To cancel a pull-down menu without making a selection, press F12 (Cancel). For example, if you select choice 9, ISPF displays the Command Table Utility pop-up, as shown in Figure 2 on page xxiv.

Note: If you entered a command on the command line prior to selecting an action bar choice, the command is processed, and the pull-down menu is never displayed. The CANCEL, END, and RETURN commands are exceptions. These three commands are not processed and the cursor is repositioned to the first input field in the panel body. If there is no input field, the cursor is repositioned under the action bar area. If you are running in GUI mode and select an action bar choice, any existing command on the command line is ignored.

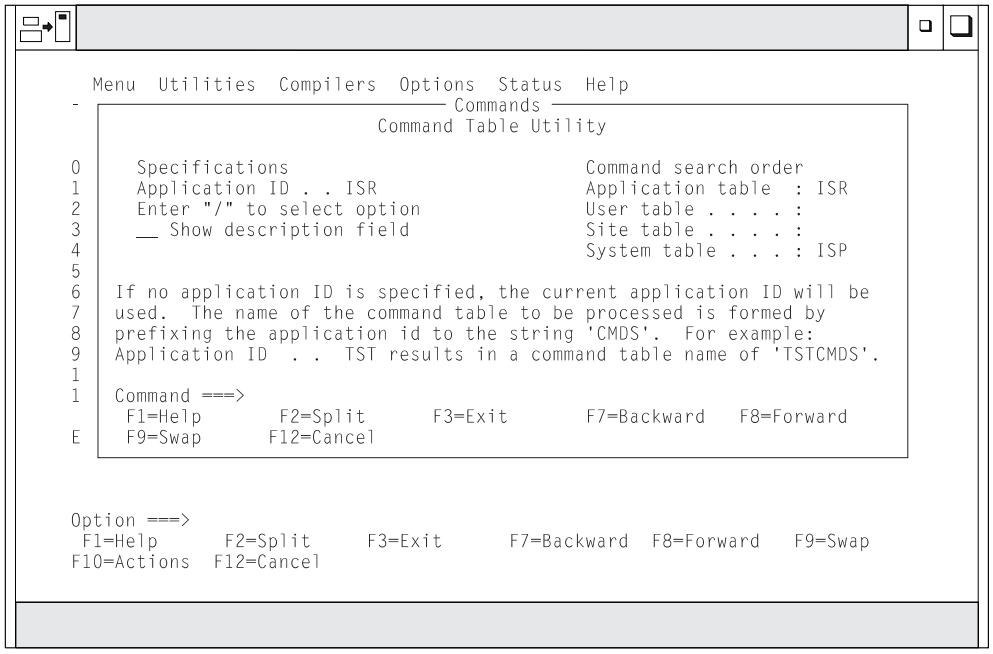
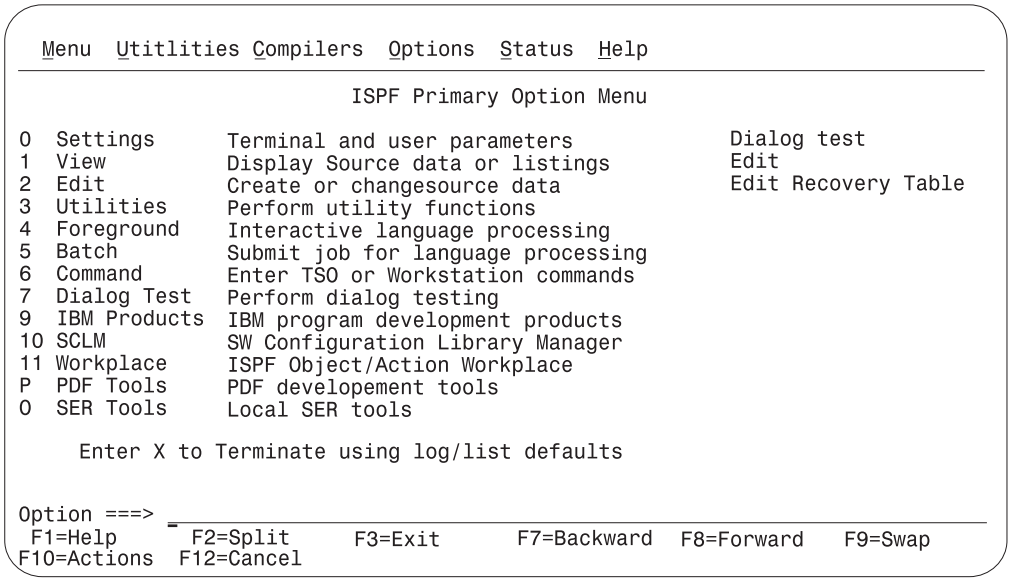


Figure 2. Pop-Up Selected from an Action Bar Pull-Down



1 Action bar. You can select any of the action bar choices and display a pull-down.

2 Options. The fields in this column are point-and-shoot text fields.

3 Dynamic status area. You can specify what you want to be displayed in this area.

Figure 3. Panel with an Action Bar and Point-and-Shoot Fields

Action Bar Choices

The action bar choices available vary from panel to panel, as do the choices available from their pull-downs. However, Menu and Utilities are basic action bar choices, and the choices on their pull-down menus are always the same.

Menu Action Bar Choice

The following choices are available from the Menu pull-down:

Settings	Displays the ISPF Settings panel
View	Displays the View Entry panel
Edit	Displays the Edit Entry panel
ISPF Command Shell	Displays the ISPF Command Shell panel
Dialog Test...	Displays the Dialog Test Primary Option panel
Other IBM Products...	Displays the Additional IBM Program Development Products panel
SCLM	Displays the SCLM Main Menu
ISPF Workplace	Displays the Workplace entry panel
Status Area...	Displays the ISPF Status panel
Exit	Exits ISPF.

Note: If a choice displays in blue (the default) with an asterisk as the first digit of the selection number (if you are running in GUI mode, the choice will be *grayed*), the choice is unavailable for one of the following reasons:

- Recursive entry is not permitted here
- The choice is the current state; for example, RefMode is currently set to Retrieve in Figure 4.

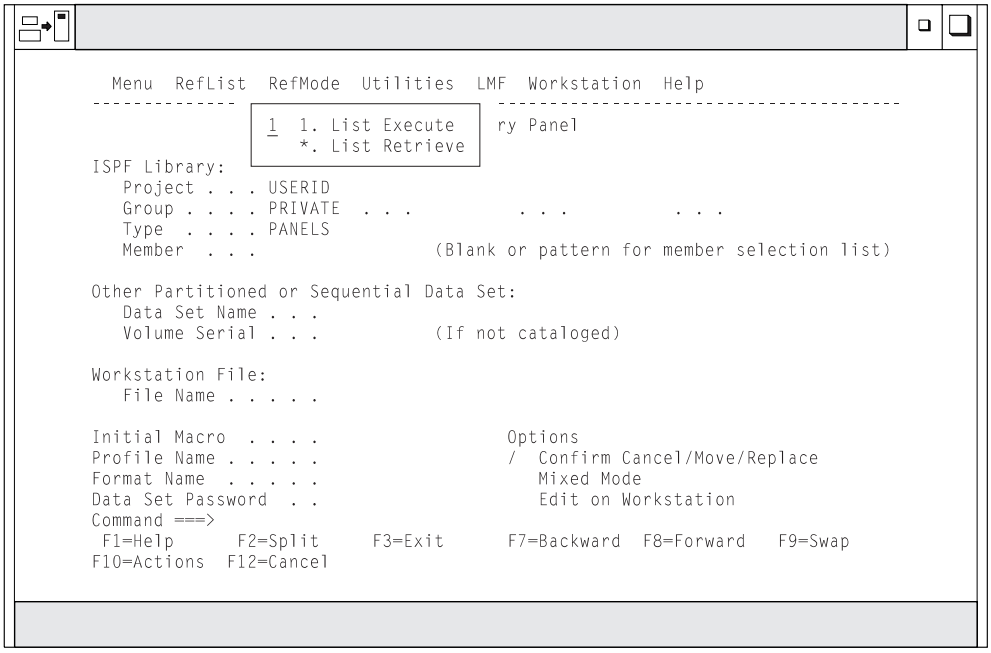


Figure 4. An Unavailable Choice on a Pull-Down

Utilities Action Bar Choice

The following choices are available from the Utilities pull-down:

Library	Displays the Library Utility panel
Data Set	Displays the Data Set Utility panel
Move/Copy	Displays the Move/Copy Utility panel

Data Set List	Displays the Data Set List Options panel
Reset Statistics	Displays the Reset ISPF Statistics panel
Hardcopy	Displays the Hardcopy Utility panel
Download...	Displays the panel that enables you to download workstation clients and other files from the host.
Outlist	Displays the Outlist Utility panel
Commands...	Displays the Command Table Utility panel
Reserved	Reserved for future use by ISPF; an unavailable choice
Format	Displays the Format Specification panel
SuperC	Displays the SuperC Utility panel
SuperCE	Displays the SuperCE Utility panel
Search-for	Displays the Search-For Utility panel.
Search-forE	Displays the Search-ForE Utility panel.

Point-and-Shoot Text Fields

Point-and-shoot text fields are cursor-sensitive; if you select a field, the action described in that field is performed. For example, if you select Option 0, Settings, in Figure 3 on page xxiv, ISPF displays the ISPF Settings panel.

Note: If you have entered a command on the command line, this command is processed before any point-and-shoot command unless you are running in GUI mode.

The cursor-sensitive portion of a field often extends past the field name. Until you are familiar with this new feature of ISPF, you might want to display these fields in reverse video (use the PSCOLOR command to set Highlight to REVERSE).

Note: You can use the Tab key to position the cursor to point-and-shoot fields by selecting the **Tab to point-and-shoot fields** option on the ISPF Settings panel (Option 0).

Function Keys

ISPF uses CUA-compliant definitions for function keys F1–F12 (except inside the Edit function). F13–F24 are the same as in ISPF Version 3. By default you see the CUA definitions because your **Primary range** field is set to 1 (Lower - 1 to 12).

To use non-CUA-compliant keys, select the **Tailor function key display** choice from the Function keys pull-down on the ISPF Settings (option 0) panel action bar. On the Tailor Function Key Definition Display panel, specify 2 (Upper - 13 to 24) in the **Primary range** field.

The following function keys help you navigate in ISPF:

F1 **Help**. Displays Help information. If you press F1 (and it is set to Help) after ISPF displays a short message, a long message displays in a pop-up window.

F2 **Split**. Divides the screen into two logical screens separated by a horizontal line or changes the location of the horizontal line.

Note: If you are running in GUI mode, each logical screen displays in a separate window.

F3 **Exit** (from a pull-down). Exits the panel underneath a pull-down.

F3 **End**. Ends the current function.

F7 **Backward**. Moves the screen up the scroll amount.

F8 **Forward**. Moves the screen down the scroll amount.

F9 **Swap**. Moves the cursor to where it was previously positioned on the other logical screen of a split-screen pair.

F10 **Actions**. Moves the cursor to the action bar. If you press F10 a second time, the cursor moves to the command line.

F12 **Cancel**. Issues the Cancel command. Use this command to remove a pull-down menu if you do not want to make a selection. F12 also moves the cursor from the action bar to the Option ==> field on the ISPF Primary Option Menu. See *ISPF Dialog Developer's Guide and Reference* for cursor-positioning rules.

F16 **Return**. Returns you to the ISPF Primary Option Menu or to the display from which you entered a nested dialog. RETURN is an ISPF system command.

Selection Fields

z/OS V1R2.0 ISPF uses the following CUA-compliant conventions for selection fields:

A single period (.)

Member lists that use a single period in the selection field recognize only a single selection. For example, within the Edit function you see this on your screen:

```
EDIT     USER1.PRIVATE.TEST               ROW 00001 of 00002
 Name         VV MM  Created   Changed  Size Init  Mod  ID
 . MEM1       01.00  94/05/12  94/07/22  40    0    0   USER1
 . MEM2       01.00  94/05/12  94/07/22  30    0    0   KEENE
```

You can select only one member to edit.

A single underscore (_)

Selection fields marked by a single underscore prompt you to use a slash (/) to select the choice. You may use any non-blank character. For example, the **Panel display CUA mode** field on the ISPF Settings panel has a single underscore for the selection field:

```
Options
  Enter "/" to select option
  _  Command line at bottom
  _  Panel display CUA mode
  _  Long message in pop-up
```

Note: If you are running in GUI mode, this type of selection field displays as a check box; that is, a square box with associated text that represents a choice. When you select a choice, a check mark (in OS/2) or an X (in Windows) appears in the check box to indicate that the choice is in effect. You can clear the check box by selecting the choice again.

An underscored field (____)

Member lists or text fields that use underscores in the selection field recognize multiple selections. For example, from the Display Data Set List Option panel, you may select multiple members for print, rename, delete, edit, browse, or view processing.

Command Nesting

Command nesting allows you to *suspend* an activity while you perform a new one rather than having to end a function to perform another function. For example, in previous versions of ISPF, if you are editing a data set and want to allocate another data set, you type `=3.2` on the command line and press Enter. ISPF *ends* your edit session before taking you to the Data Set Utility panel. When you have allocated the data set and want to return to your edit session, you type `=2` and press Enter; ISPF returns you to the Edit Entry Panel. With z/OS V1R2.0 ISPF, from your edit session, select the Data set choice from the Utilities pull-down on the Edit panel action bar. ISPF suspends your edit session and displays the Data Set Utility panel. When you have allocated the new data set and end the function, z/OS V1R2.0 ISPF returns you directly to your edit session rather than to the Edit Entry Panel.

Elements and Features in z/OS

You can use the following table to see the relationship of a product you are familiar with and how it is referred to in z/OS Version 1 Release 2.0. z/OS V1R2.0 is made up of elements and features that contain function at or beyond the release level of the products listed in the following table. The table gives the name and level of each product on which a z/OS element or feature is based, identifies the z/OS name of the element or feature, and indicates whether it is part of the base or optional. For more compatibility information about z/OS elements see *z/OS Planning for Installation, GC28-1726*

Product Name and Level	**Name in z/OS**	**Base or Optional**
BookManager BUILD/MVS V1R3	BookManager BUILD	optional
BookManager READ/MVS V1R3	BookManager READ	base
MVS/Bulk Data Transfer V2	Bulk Data Transfer (BDT)	base
MVS/Bulk Data Transfer File-to-File V2	Bulk Data Transfer (BDT) File-to-File	optional
MVS/Bulk Data Transfer SNA NJE V2	Bulk Data Transfer (BDT) SNA NJE	optional
IBM OS/390 C/C++ V1R2	C/C++	optional
DFSMSdfp V1R3	DFSMSdfp	base
DFSMSdss	DFSMSdss	optional
DFSMShsm	DFSMShsm	optional
DFSMSrmm	DFSMSrmm	optional
DFSMS/MVS Network File System V1R3	DFSMS/MVS Network File System	base
DFSORT R13	DFSORT	optional
EREP MVS V3R5	EREP	base
FFST/MVS V1R2	FFST/MVS	base
GDDM/MVS V3R2 • GDDM-OS/2 LINK • GDDM-PCLK	GDDM	base
GDDM-PGF V2R1.3	GDDM-PGF	optional
GDDM-REXX/MVS V3R2	GDDM-REXX	optional
IBM High Level Assembler for MVS & VM & VSE V1R2	High Level Assembler	base
IBM High Level Assembler Toolkit	High Level Assembler Toolkit	optional
ICKDSF R16	ICKDSF	base
ISPF	ISPF	base
Language Environment for MVS & VM V1R5	Language Environment	base
Language Environment V1R5 Data Decryption	Language Environment Data Decryption	optional

© Copyright IBM Corp. 1989, 2001

Product Name and Level	Name in z/OS	Base or Optional
MVS/ESA SP V5R2.2		
BCP	BCP or MVS	base
ESCON Director Support	ESCON Director Support	base
Hardware Configuration Definition (HCD)	Hardware Configuration Definition (HCD)	base base
JES2 V5R2.0	JES2	optional
JES3 V5R2.1	JES3	base
LANRES/MVS V1R3.1	LANRES	base
IBM LAN Server for MVS V1R1	LAN Server	base
MICR/OCR Support	MICR/OCR Support	base
OS/390 UNIX System Services	OS/390 UNIX System Services	base
OS/390 UNIX Application Services	OS/390 UNIX Application Services	base
OS/390 UNIX DCE Base Services (OSF DCE level 1.1)	OS/390 UNIX DCE Base Services	base
OS/390 UNIX DCE Distributed File Services (DFS) (OSF DCE level 1.1)	OS/390 UNIX DCE Distributed File Services (DFS)	optional
OS/390 UNIX DCE User Data Privacy	OS/390 UNIX DCE User Data Privacy	optional
SOMobjects Application Development Environment (ADE) V1R1	SOMobjects Application Development Environment (ADE)	base
SOMobjects Runtime Library (RTL)	SOMobjects Runtime Library (RTL)	base
SOMobjects service classes	SOMobjects service classes	
Open Systems Adapter Support Facility (OSA/SF) R1	Open Systems Adapter Support Facility (OSA/SF)	base
MVS/ESA RMF V5R2	RMF	optional
OS/390 Security Server	Resource Access Control Facility (RACF) • DCE Security Server • OS/390 Firewall Technologies • Lightweight Directory Access Protocol (LDAP) Client and Server • Open Cryptographic Enhanced Plug-ins (OCEP)	optional
SDSF V1R6	SDSF	optional
SMP/E	SMP/E	base
	Softcopy Print	base
SystemView for MVS Base	SystemView for MVS Base	base
IBM TCP/IP V3R1 • TCP/IP CICS Sockets • TCP/IP IMS Sockets • TCP/IP Kerberos • TCP/IP Network Print Facility (NPF) • TCP/IP OS/390 Communications Service IP Applications • TCP/IP OS/2 Offload	TCP/IP • TCP/IP CICS Sockets • TCP/IP IMS Sockets • TCP/IP Kerberos • TCP/IP Network Print Facility (NPF) • TCP/IP OS/390 Communications Service IP Applications • TCP/IP OS/2 Offload	base • optional • optional • optional • optional • optional • optional
TIOC R1	TIOC	base
Time Sharing Option Extensions (TSO/E) V2R5	TSO/E	base

Product Name and Level	Name in z/OS	Base or Optional
VisualLift for MVS V1R1.1	• VisualLift Run-Time Environment (RTE) • VisualLift Application Development Environment (ADE)	• base • optional
VTAM V4R3 with the AnyNet feature	VTAM	base
3270 PC File Transfer Program V1R1.1	3270 PC File Transfer Program	base

Part 1. Guide to the Dialog Tag Language (DTL)

This section contains the following chapters:

Chapter 1. Introduction to the Dialog Tag Language (DTL) contains an introduction to the Dialog Tag Language (DTL) and descriptions of the dialog elements you define with the Dialog Tag Language. A brief description of the ISPF Conversion Utility is also included.

Chapter 2. How to Use the Dialog Tag Language (DTL) explains the syntax conventions of the Dialog Tag Language.

Chapter 3. Getting Started: Designing Application Panels tells you how to design application panels.

Chapter 4. Variables and Variable Classes tells you how to declare variables, define variable classes, and validate variables.

Chapter 5. Application Panel Fields tells you how to define interactive fields for application panels.

Chapter 6. Information Regions and Help Panels tells you how to define information regions and help panels.

Chapter 7. Messages tells you how to define messages.

Chapter 8. The Application Command Table tells you how to define application commands and the application command table.

Chapter 9. Defining Key Mapping Lists tells you how to define key mapping lists.

Chapter 10. Using the Conversion Utility describes the steps involved in converting your DTL source files into ISPF panel language source format or executable pre-processed ISPF format.

© Copyright IBM Corp. 1989, 2001

Chapter 1. Introduction to the Dialog Tag Language (DTL)

This book introduces you to the Dialog Tag Language (DTL), a tag-based language used to define many of the elements that make up the type of application known as a *dialog*, the communication between a person and a computer. We tell you how to define these elements using DTL and how to prepare them for use in your ISPF applications.

The elements you produce with DTL are used by ISPF as the user interface for your ISPF applications. The programs you write using ISPF services and a programming language use the dialog elements you create for an application.

This chapter provides an overview of DTL, and the dialog elements you create with DTL. We introduce you to DTL by discussing the following main topics:

- **Why the Dialog Tag Language?**

  This section talks about why you would want to use DTL to create elements for ISPF applications.
- **What is the Dialog Tag Language?**

  This section explains what the Dialog Tag Language is and how it works.
- **Dialog elements**

  This section provides descriptions and examples of the dialog elements. These elements are:
  - Application panels
  - Help panels
  - Messages
  - An application command table
  - Key mapping lists.
- **Variables and variable classes**

  This section discusses the definition of variables you include in dialog element definitions.
- **What is the ISPF Conversion Utility?**

  This section provides a description of the conversion utility, the compiler you use to convert your DTL source files for use by ISPF.

Why the Dialog Tag Language (DTL)?

If you are already familiar with a tag-based markup language, such as IBM BookMaster*, you will find that DTL is very similar. We created DTL for many of the same reasons that we created BookMaster:

- Markup tags are easy to use. Because tag names are short and relate directly to the structure of the dialog elements, they are also easy to remember.
- DTL lends flexibility to application development. Panels can be quickly changed without your having to tediously line up text and fields. This gives you greater control over application development and updates.
- DTL provides consistency when many programmers are working on the same application, or when programmers who are new to your company must update existing applications. Since each programmer is using the same tags, only minor adjustments may be needed to achieve complete uniformity.

© Copyright IBM Corp. 1989, 2001

- DTL techniques improve the way in which interactive programs, like ISPF applications, are developed. The language concentrates on the role of the various elements and their interrelationships, and ISPF takes care of their form and appearance at run time.
- DTL also enforces some formatting rules defined by the Systems Application Architecture* Common User Access (CUA), so you do not have to be familiar with all of the CUA formatting rules. Therefore, the CUA skills required by programmers who are developing CUA-conforming applications are significantly reduced.
- DTL enables National Language Support (NLS) and the conversion utility provides NLS translations for certain key words.

In other words, if you are looking for an application development and maintenance system that is sophisticated, flexible, and easy to use, that's DTL.

Examples of DTL usage by ISPF have been shipped in data set 'ISP.SISPGxxx', where xxx is a standard ISPF language identifier. Consult your ISPF system administrator for the actual location of these examples.

What Is the Dialog Tag Language?

In Why the Dialog Tag Language (DTL)? we referred to DTL as a tag-based markup language that is similar to IBM BookMaster. The two have much in common. For example, *markup* is a term that is usually associated with documentation. It is an old typesetter's term that formerly meant the instructions with which a document was "marked up" to show how the document should be set in type.

Today, this definition has been expanded to include information that is added to a document to enable a person or system to process it. Just as markup information can describe a document's characteristics or the processing to be performed, it can also describe the characteristics or processing related to dialog elements. This is where the tags come in.

We call DTL a *tag-based* markup language simply because the markup consists of tags that determine not only what each element is, but also how it is processed. To convert the dialog elements into a format that is usable by ISPF, you must convert them to ISPF elements with ISPDTLC, the ISPF conversion utility. (See "What Is the ISPF Conversion Utility?" on page 9 for more information.)

Another thing that DTL and BookMaster have in common, of course, is the tags themselves, which have the following similarities:
- They are very short and easy to remember.
- They are often accompanied by text.
- Many DTL tags are almost identical to corresponding BookMaster tags.

These are all reasons that familiarity with BookMaster will make it easy to learn DTL. For example, we created the preceding bulleted, or *unordered* list by editing a file using an editor and typing the following:

```
:ul compact.
  :li.They are very short and easy to remember.
  :li.They are often accompanied by text.
  :li.Many DTL tags are almost identical to corresponding BookMaster tags.
:eul.
```

You could create a similar list for an information panel by using the following DTL tags with the same text:

```
<ul compact>
  <li>They are very short and easy to remember.
  <li>They are often accompanied by text.
  <li>Many DTL tags are almost identical to corresponding BookMaster tags.
</ul>
```

Here, the <ul> and tags, respectively, begin and end the unordered list. This type of list is called an unordered list because the list items are not numbered. The individual list items are defined by the <li> tags and consist of the accompanying text.

As you can see from the preceding example, DTL tags act as control words that specify how the text of source files is interpreted by the conversion utility. This concept is based on the Standard Generalized Markup Language (SGML), which is a standard of the International Standards Organization (ISO). The conventions of the Dialog Tag Language are based on the SGML standard.

After you are finished marking up a source file, you use the conversion utility to convert the file into a format usable by your ISPF application. In addition to processing the file, the conversion utility also checks and verifies the syntax of your markup, and notifies you of any errors. After conversion, the elements you defined in your source file are stored within ISPF libraries.

You can use ISPF dialog test facilities to display application panels and messages after they have been converted. Displaying your panels is a good idea to make sure they format properly.

You should now have a basic understanding of DTL and how it works. The next section builds on this understanding by describing the types of elements that you can define with DTL.

Dialog Elements

This section provides a descriptive overview of the dialog elements you can create for an ISPF application. These elements include:

- Application panels
- Help panels
- Messages
- An application command table
- Key mapping lists.

Application Panels

Application panels are the primary element of the user interface for an application. They allow users to interact with your application through the use of data fields, selection fields, and other interactive fields. Application panels appear in primary and pop-up windows.

Figure 5 on page 6 shows a full-screen application panel. Following that is a list of the elements that make up an application panel.

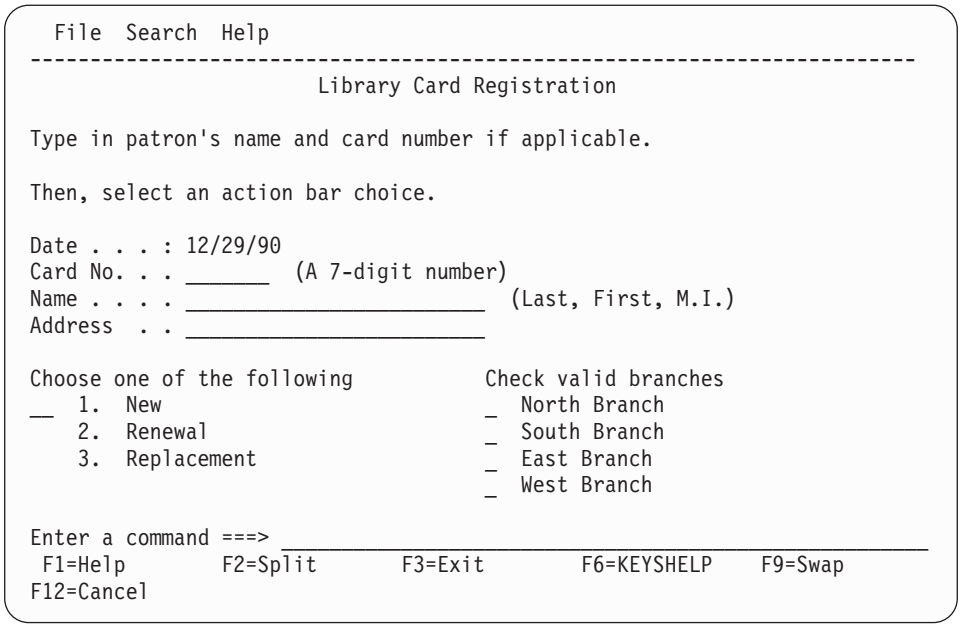


Figure 5. Application Panel

Application Panel Elements

Action bar

The action bar appears in the top portion of the panel. It contains keyword choices that provide users access to available actions for the current panel. When the user selects an action bar choice, a pull-down containing choices appears directly below the action bar choice.

Panel title

The panel title appears below the action bar.

Panel body

The panel body serves as the main work area of the panel. The panel body contains the input and output fields, selection fields, and other text.

Additionally, the panel body can contain optional top and bottom instructions, which provide instructional text to the user. Top instructions appear below the panel title and above the interactive fields on the panel. Top instructions tell the user how to interact with the panel. Bottom instructions appear below the interactive fields on the panel. Bottom instructions tell the user how to interact with the panel, or how to continue with the application.

Message area

The message area appears below the panel body.[1] ISPF uses the message area (or message pop-ups) to display messages to users while they are working in the panel.

1. The message area and the command area for panels defined with DTL appear at the bottom of the panel if the user has selected the Command line at bottom option on the ISPF Settings panel, or the application has set ZPLACE to BOTTOM. For additional information on placement options, refer to the discussion of the ISPF Settings panel in the *ISPF User's Guide*.

Command area
The optional command area (or command line), which appears at the bottom of the panel, consists of two components: the command field prompt and the command entry field. [1] Application users can use the command entry field to enter commands or requests to the ISPF application.

Function key area
The optional function key area, which appears at the bottom of the panel immediately below the command area (if one is defined), contains the key assignments for dialog actions valid for the application panel. The user can request that function keys not be displayed.

"Chapter 3. Getting Started: Designing Application Panels" on page 29 tells you how to define application panels and panel elements.

Help Panels

Help panels appear in pop-up windows in response to user requests for assistance during ISPF application sessions. ISPF processes these help requests and displays the help panels.

Using DTL, you can create help panels that provide help for:
- An entire application panel (extended help or panel help)
- A specific field on an application panel (contextual help or field help)
- Messages (message help)
- The function key area (keys help).

Figure 6 illustrates a help panel. Following that is a list that defines each of the elements that make up a help panel.

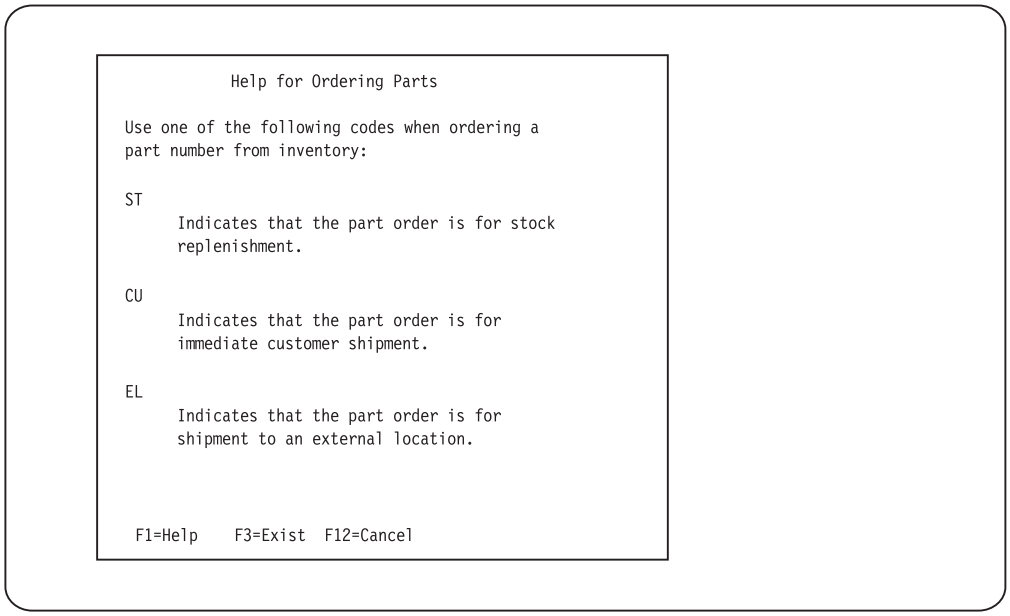


Figure 6. Help Panel

Help Panel Elements

Help panel title
The help panel title appears at the topmost portion of the panel, followed

by a blank line separating the panel title from the panel body. If the help panel text exceeds the defined depth of the help panel, a scrolling indicator appears in the right margin of the blank line following the panel title.

Help panel body
The help panel body contains the text of the help panel.

Text within the help panel is protected, which means that the user cannot interact with the text. You define this static text within an *information region* in the panel definition.

Function key area
If you are creating a help panel that does not end with a scrollable area, note that ISPF reserves 4 lines at the bottom of the panel for function keys. The display of keys in the function key area is controlled by the user through the ISPF FKA command.

"Chapter 6. Information Regions and Help Panels" on page 115 tells you how to define help panels and information regions.

Messages

You can use DTL to define messages that display in response to a user request or action, or that provide additional information. Messages can confirm a user-requested action, report an error in user input, or notify the user of an error or exception condition. Figure 7 illustrates a message displayed in the message area of an application panel (highlighting added).
The messages you define for an application are stored within *message members*. You

```
  File  Search  Help
------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number if applicable.

Then, select an action bar choice.

Date . . . : 12/29/90
Card No. . . _______  (A 7-digit number)
Name . . . . _____________________________  (Last, First, M.I.)
Address  . . _____________________________

Choose one of the following           Check valid branches
__  1.  New                           _  North Branch
    2.  Renewal                       _  South Branch
    3.  Replacement                   _  East Branch
                                      _  West Branch
You must type your name in the Name field.
Enter a command ===> ___________________________________________________________
 F1=Help       F2=Split       F3=Exit        F6=KEYSHELP    F9=Swap
F12=Cancel
```

Figure 7. Message Displayed in Message Area

use DTL to define the messages and message members.

"Chapter 7. Messages" on page 159 provides a complete description of defining messages and message members.

Application Command Table

You can use DTL to define commands that perform actions requested by the user. The valid commands for an application are defined and stored within an internal application command table. You can define only one command table for an application.

Valid commands include those assigned to pull-down choices, function keys, and commands entered in command entry fields.

"Chapter 8. The Application Command Table" on page 165 tells you how to define commands and application command tables.

Key Mapping Lists

The key assignments that are active for an application are defined and stored within key mapping lists. These key assignments allow the user to request commands and other actions through the use of function keys. Key assignments for your application are displayed in the function key area of application panels.

"Chapter 9. Defining Key Mapping Lists" on page 171 tells you how to define key assignments and key mapping lists.

Variables and Variable Classes

Variables are used to communicate information between an application and the user. Each variable you define for a DTL-defined dialog element can be *declared*, or identified, within a variable list. In addition, each variable can be associated with a variable class that defines its type and length characteristics. The variable class can also be used to define translations and validity checks that are used when a value is displayed on a panel or entered by a user.

"Chapter 4. Variables and Variable Classes" on page 61 tells you how to declare variables and define variable classes.

What Is the ISPF Conversion Utility?

ISPDTLC is the ISPF conversion tool that converts Dialog Tag Language (DTL) source files to ISPF panel language source format or executable preprocessed ISPF format. ISPF provides you with an invocation panel that allows you to specify a number of options for the conversion, or you can use conversion utility command syntax from the command line of your terminal. "Chapter 10. Using the Conversion Utility" on page 175 provides a complete description of both methods.

Chapter 2. How to Use the Dialog Tag Language (DTL)

This chapter describes the tag syntax conventions and mark-up declarations of the Dialog Tag Language (DTL). It also explains how to use the DTL to create dialog element source files for your ISPF applications.

The markup style of DTL is based on the International Standards Organization (ISO) Standard Generalized Markup Language (SGML). Markup languages allow you to specify, through the use of tags, how the text of a file is to be formatted for use by an application. Because DTL is a markup language, you must follow special rules and conventions when using it to define dialog elements.

Syntax Conventions

The DTL tags act as control words that determine how the text in the source files is used. Each tag is enclosed within a set of delimiter symbols that distinguish the tag as a control word (as opposed to general text).*Start* tags, which initiate text interpretation,are preceded by the start tag open delimiter (<) and followed by the close delimiter (>). *End* tags, which explicitly end text interpretation,are preceded by the end tag open delimiter (</) and followed by the close delimiter (>).

For example, the DTL tags used for defining the beginning and end of an application panel are the PANEL tag and its matching end tag, which look like this:[2]

```
<panel>
</panel>
```

DTL tags are free-form. Indentation of nested tags can be helpful for DTL source file readability.

All of the text that you define between a start and end tag is the tag *definition*. The DTL tag data extends to the right boundary of the source file. Therefore, DTL source files cannot contain line sequence numbers. The characteristics of the tag determine how the text or other tags coded within the tag definition will be formatted.

Attributes and Values

Many DTL start tags contain *attributes* and *values* that define various physical and operating characteristics of the dialog elements. While most attributes and values are optional, or contain default settings, some are required.

For example, the PANEL tag has a required NAME attribute that must be specified to identify the panel. The value you assign to the NAME attribute must be unique for each panel in a source file. This PANEL tag has the NAME value “panel1”:

```
<panel name=panel1>
</panel>
```

2. End tags are not required for all of the DTL tags. In many cases, the tag is implicitly ended by other start tags. For this reason, we don’t use optional end tags in the markup examples in this book. “Chapter 13. Tag Reference” on page 211 contains a detailed description of each of the DTL tags, including which tags require an end tag.

© Copyright IBM Corp. 1989, 2001

The PANEL tag also has two optional attributes, DEPTH and WIDTH, whose values specify the dimensions of the panel. For these types of attributes, you specify a numeric value.

```
<panel name=panel1 depth=20 width=40>
</panel>
```

Values for some of the tag attributes are predetermined; that is, you can choose from one of a number of keyword values for the tag. For example, the FIG (figure) tag has a FRAME attribute that specifies the top and bottom borders of the figure. The value you assign to the FRAME attribute can be either RULE, which produces a visible border above and below the figure, or NONE, which results in a figure without a border. No other value is acceptable for the FRAME attribute.

RULE is the default value, which means that the figure formats with visible borders if you do not specify the FRAME attribute.

The markup for a figure without ruled borders looks like this:

```
<fig frame=none>
</fig>
```

When coding values for attributes, you must use single or double quotes to enclose values that contain characters other than A–Z, a–z, 0–9, a hyphen (–), or a period (.).

For example, the value assigned to the TYPE attribute of this VARCLASS tag contains a blank, so the value must be enclosed in quotes:

```
<varclass name=boolean type='char 1'>
```

Some attributes can be assigned either a specific value, such as a number or a character string, or a variable name. To distinguish a variable name from a specific value, precede the variable name with a percent (%) sign. This convention is called *% notation*.The percent sign distinguishes the variable name from a specific value. To specify a string that begins with a %, you must code an additional % before the string to distinguish it from a variable name. (For example, to specify the string %“abc”, code “%%abc”).

The ACTION attribute in the following example uses % notation to specify a variable named “varname”:

```
<cmdact action='%varname'>
```

The length of any attribute value is limited to 253 characters, unless stated otherwise. This includes the lengths of any entity references that are a part of the value.

Generally, you can code tags, attributes, and values in uppercase, lowercase, or mixed case; the results are always the same regardless of case. The conventions you must follow for case-sensitive processing for each tag are described in “Chapter 13. Tag Reference” on page 211.

Tag Text

The content or text of a tag is coded immediately following the start tag. This is the actual text that is subject to formatting and translation. The text is processed according to the type of tag it follows.

For example, the text following this P (paragraph) tag is the actual text that appears in the panel after formatting:

```
<p>The copy command allows you to copy single or multiple forms.
```

Because the tag text is processed according to the tag characteristics, not the way it is written in the source file, the paragraph could also be marked up using more than one line, like this:

```
<p>
The copy command allows you to
copy single or multiple forms.
```

The formatted result is the same in either case.

In most cases, there is no limit to the amount of text you can code. However, keep in mind that the text of some tags, such as the title of a PANEL tag, should be limited because of size constraints of the panel they are coded within. "Chapter 13. Tag Reference" on page 211 describes text length restrictions (if they exist) for each of the tags.

In most cases, multiple lines of text are concatenated. Concatenation, leading blanks, and trailing blanks are processed as follows:

- Leading and trailing blanks between lines of text are not preserved. Instead, they are compressed to a single blank when the lines are concatenated.
- The first line of tag text may start on the same line as the start tag, or on the next line. The formatted result is the same.

The text of some tags, such as the FIG, LINES, and XMP tag, allow you to control where lines break. That is, within the range of the tag, each output line is ended at the same point that you ended the input line. With these tags, multiple lines are not concatenated, and all blanks are preserved.

Text Formatting

ISPF determines if the text is to be formatted according to English rules or Asian rules, based on the language specified on the conversion utility invocation. If the language is JAPANESE, CHINESET, CHINESES, or KOREAN, ISPF will use the Japanese, Traditional Chinese, Simplified Chinese, or Korean text formatting rules, respectively. If JAPANESE language is specified and the KANA option is also specified, ISPF uses the Japanese Katakana formatting rules. Otherwise, the English formatting rules are used.

English Rules for Text Formatting

Text exceeding the width of the available panel space is wrapped to the next line. The text is split at blanks. However, if any word exceeds the panel space, then the word splits and continues on the next line.

Asian Rules for Text Formatting

Some characters should not be placed at the beginning of a line, and some should not be placed at the end of a line. These beginning-and-ending inhibited characters are different among the languages but the required process is the same. Thus, ISPF uses the same text formatting process for these Asian languages, but uses a different beginning-and-ending inhibitor character table for each of the languages.

The text is first split into *words*. An SBCS *word* is delimited by blanks, or SO/SI characters. Then any beginning inhibitors are stripped from the beginning of the

word and treated as separate words, and any ending inhibitors are stripped from the end of the word and treated as separate words.

Adjoining DBCS alphanumeric characters (that is, Ward 42 characters) are treated as one DBCS *word*. Then any beginning inhibitors are stripped from the beginning of the word and treated as separate words, and any ending inhibitors are stripped from the end of the word and treated as separate words. All other non-Ward 42 double-byte characters are treated as separate DBCS *words*.

If a word exceeds the available panel space, then the word splits and continues on the next line. If the text consists of mixed data and does not fit in one line within the specified width, the first position will always be reserved for a SO character (if first word is double-byte) or for a blank (if the first word is single-byte). This allows the text to be aligned properly.

Words that exceed the width of the available panel space are wrapped to the next line according to following rules:

```
      ... CE–1 CE
CB CB+1 ...
```

CE–1	CE	CB	CB+1	Process
any	B,X	B	X,E	Backward
E	E	X,B	X,E	Backward
X,B	E	any	any	Forward
X,B	X	B	B	Forward
any other				No process

Figure 8. Text Formatting Rules

Where:

CE-1 and CE	Last two words that fit on line
CB and CB+1	First two words on next line
E	Ending inhibitor
B	Beginning inhibitor
X	Neither
Forward	Move CE to next line
Backward	Move CB to previous line
No process	Split as is

Note: If words CE or CB are single-byte words and are more than 1 character, or if CE or CB are double-byte words and are more than 1 double-byte character, then no special processing is used; the line is split as is.

When your panel contains several successive lines of mixed data from different tags, the alignment of a short text string can appear to be shifted 1 byte further left than the surrounding text. This occurs because a text string that fits on one line does not have the leading position reserved for the SO character to use as many positions on the screen as possible.

You can control the alignment of successive lines of mixed data by adding a string of DBCS blanks to the end of a short text string. This forces the SO character position to be reserved during formatting.

SBCS and DBCS blanks that end or begin a line will be deleted.

Nesting tags

It is often necessary to code certain tags (and their text) within the definition of other tags (between the start and end tags). This is called *nesting*.

A good example of nesting is the relationship between the DL (definition list) tag, the DT (definition term) tag, and the DD (definition description) tag. The DL tag specifies a definition list and the DT and DD tags specify the terms and descriptions of the items within the definition list. Consequently, the DT and DD tags must be nested within a DL tag and its matching end tag if the list is to format properly.

Here's an example:[3]

```
<dl>
  <dt>This is a definition term.
  <dd>This is a definition description.
  <dt>Another term.
  <dd>Another description.
</dl>
```

There are several tags that must be nested within the actual text of another start tag. These tags serve to identify a condition for the text. In this example, the nested CMD tag follows the CMDTBL start tag and precedes the CMDTBL end tag. The T (truncation) tag nested within the text of the CMD tag provides truncation of the command text.

```
<CMDTBL APPLID=conv>
  <CMD NAME=delete>Del<T>ete
    <CMDACT ACTION=setverb>
</CMDTBL>
```

Nesting tags can take on many different forms and can be complex. For example, some tags allow multiple tags or multiple occurrences of the same tag to be nested, while other tags do not allow nesting of any tags. You can also nest levels of certain tags, that is, nested tags within other nested tags. Additionally, in many instances, you must nest certain tags within other tags. The tag descriptions in "Chapter 13. Tag Reference" on page 211 describe the allowed and required conditions for nesting each of the DTL tags.

Including Comments in the Generated Panel or Message Member

You can use the COMMENT tag to add comments to the generated panel or message member file. The TYPE attribute specifies the panel section for the comment. TYPE = END is automatically used for message member processing. You provide the comment text in a manner similar to the paragraph tag. ISPDTLC flows the text to a width of 66 bytes and adds "/* " before and " */" after each resulting comment line.

Including Copyright Statements in the Generated Panel or Message Member

You can use the COPYR tag to add a copyright statement to the generated panel or message member. The copyright statement is placed in the panel immediately following the)END panel section line, or immediately following the last message in the message member. The text of the COPYR tag is limited to 66 bytes. ISPDTLC adds "/* " before and " */" after the copyright text. Each COPYR tag adds one line to the generated panel.

3. Although it isn't required, we indented the nested tags in this example to illustrate nesting levels. You can also do this in your own source files.

Markup Declarations

In addition to tag markup, you can also include *markup declarations* in your source files to define other, related information. Markup declarations are control statements that specify how other markup (such as tags) within a source file is to be interpreted.

For example, in order for the compiler to recognize your source files as being intended for DTL conversion to ISPF elements, you must include a *document type* declaration at the beginning of each source file.

Like tags, markup declarations must be enclosed within a set of delimiter symbols so the compiler can distinguish the declaration as a control statement. All markup declarations are preceded by the <! symbol and followed by the > symbol. (For NLS users of DTL the <! symbol can be replaced with the <: symbol.)

DTL supports three types of markup declarations:
- Document type declarations
- Comments
- Entity declarations.

Declaring the Document Type

Before you can convert a source file that contains the tag markup for dialog elements, you must declare the *document type*. Do this by coding the DOCTYPE declaration at the beginning of the source file. The DOCTYPE declaration looks like this:

```
<!doctype dm system>
```

Where:

<!	Begins the markup declaration
DOCTYPE	Identifies the declaration as a document type declaration
DM	Specifies that the source file contains tags used to define dialog elements for a Dialog Manager application
SYSTEM	Indicates that the syntax rules for defining elements are contained in an external file
>	Closes the markup declaration.

External files that are embedded (through the use of entity declarations) within the source file intended for conversion cannot contain a DOCTYPE declaration. They are converted using the DOCTYPE declaration of the source file they are embedded within. For more information on entity declarations and embedding external files within source files, see "Defining Entities and Parameter Entities" on page 18.

Including Comments in Your Markup

If you want to include notes, reminders, or other text that you don't want processed in your source files, you can insert them as comments, and the conversion utility ignores them.

Note: You cannot place comments within any of the DTL tags. A comment placed within a start or end tag causes the tag to end, and the text following the comment is treated as part of the tag content.

Like document type declarations, comments must be enclosed within markup declaration delimiters (<! >). However, you must also delimit comments within markup declarations by preceding and following a comment with two dashes (--), like this:

```
<!-- This is the text of the comment -->
```

Because the dashes act as comment delimiters,you can use them in any markup declaration. For example, you can include a comment within a DOCTYPE declaration:

```
<!doctype dm system -- DECLARE DOCUMENT TYPE -->
```

The following comment will generate a warning message because the second set of dashes is interpreted as the end of a comment and the text “Provides help for ordering” is treated as an additional markup declaration:

```
<!-- Panel DMH022 -- Provides help for ordering -->
```

If you delete one of the dashes in the second set of dashes, or use another symbol, no error will occur.

```
<!-- Panel DMH022 - Provides help for ordering -->
```

This block comment will produce a warning message because of the odd number of dashes in the first and last lines of the block:

```
<!----------------------------------------->
<!--This source file contains all of the -->
<!-- help panels for the application     -->
<!----------------------------------------->
```

We could avoid this problem by using a different symbol between the comment dashes, like this:

```
<!--***************************************-->
<!--This source file contains all of the -->
<!-- help panels for the application     -->
<!--***************************************-->
```

ISPDTLC accepts comments which start with the 4 characters “<!--” and end with the 3 characters “-->”. The minimum valid comment is 7 characters (“<!---->”).

You cannot nest comments within other comments. You can, however, code multiple comments within a markup declaration, like this:

```
<!-- Here a comment --
-- THERE A COMMENT --
-- Everywhere a comment, COMMENT-->
```

As you can see, each of the comments begin and end correctly with the comment delimiters.

You can use comment delimiters to temporarily ignore multiple lines (or a block) of DTL source text. The block of text might include one or more DTL tags. To *comment out* a block of text, place an ***open comment*** delimiter before the first line of the text, and a ***close comment*** delimiter after the last line of text. For example:

```
<!--
<p> This is a multiple line of text block
<p> It is commented out for compile purposes
-->
```

When commenting out multiple lines of DTL source, use the **MCOMMENT** compiler option when coding the **ISPDTLC** invocation syntax, or select the ***Process multiple line comment blocks*** option on the ISPDTLC invocation panel.

Defining Entities and Parameter Entities

You can define, or *declare* frequently used words, phrases, and longer character strings in your source file as *entities* or *parameter entities* that represent text in the source file. You declare them within the DOCTYPE statement of your source file. After you declare them, you refer to the names of the entities in place of the word or phrase in the text. This saves you time when marking up your text, and allows you to globally change the defined words or phrases in one place in the source file.

You can use entities and parameter entities for the following purposes:

- To replace single characters in text that are considered special characters. This can include characters not available on a particular keyboard, or characters that have special meaning to the compiler, such as the tag start delimiter (<), that you want to treat as normal text.

  DTL provides you with a set of predefined single-character entities. See "Predefined Entities" on page 25 for a list of these entities.
- To replace strings of text, such as words, phrases, and longer text strings used frequently in the source file text.
- To embed entire files in a source file. This is useful for breaking up a source file into smaller, more manageable files, and for declaring entities that are shared by different source files.

When you refer to an entity in the text of a source file, you must precede the entity reference with an ampersand (&) and follow it with a semicolon (;) or a blank space. The text defined by the entity replaces the entity reference in the formatted text.

Entities

Entities are symbolic statements that represent text strings in a source file. Like other markup declarations, entity declarations must be enclosed within markup declaration delimiters (<! >). In addition, you must place entity declarations within the *declaration subset* of the DOCTYPE statement.

The declaration subset is delimited by left and right brackets ([ ]) or parentheses () and is coded within the DOCTYPE statement. If left and right brackets are coded, they must have the hex values of 'AD' and 'BD' respectively.

Within the markup declaration delimiters, you declare the entity with the term "entity", the name you are assigning to the entity, and the text string the name represents. The text string of the entity must be enclosed in single or double quotes.

```
<!doctype dm system (
<!entity name "text string">
)>
```

Entity names must have the following characteristics:

- 1–17 characters
- The first character must be alphabetic (A–Z, a–z, @, #, or $)
- Remaining characters, if any, can be A–Z, a–z, @, #, $, 0–9, or _
- Entity names are case-sensitive.
- Entity names of more than 8 bytes must contain at least 1 underscore character.

This example declares an entity named “guar” for the phrase “full, unconditional, money-back guarantee”:

```
<!doctype dm system [
<!entity guar "full, unconditional, money-back guarantee">
]>
```

Now that we’ve declared the entity, we can use the entity name in our source file text instead of the entire text string. To specify an entity name in text, you must precede the name with an ampersand (&) and follow it with a semicolon (;) or a blank, as we did in this panel text:

```
<!doctype dm system [
<!entity guar "full, unconditional, money-back guarantee">
]>
<panel name=widget21 width=40>Widgets
  <area>
    <info width=38>
      <p>You'll love the wide selection of merchandise
      in our Widgets department.
      <p>And, like all of our merchandise, Widgets come
      with our &guar;.
    </info>
  </area>
</panel>
```

As long as we declared the entity properly, the compiler recognizes the entity reference in the source file and replaces it with the text of the entity declaration. Figure 9 shows the result.

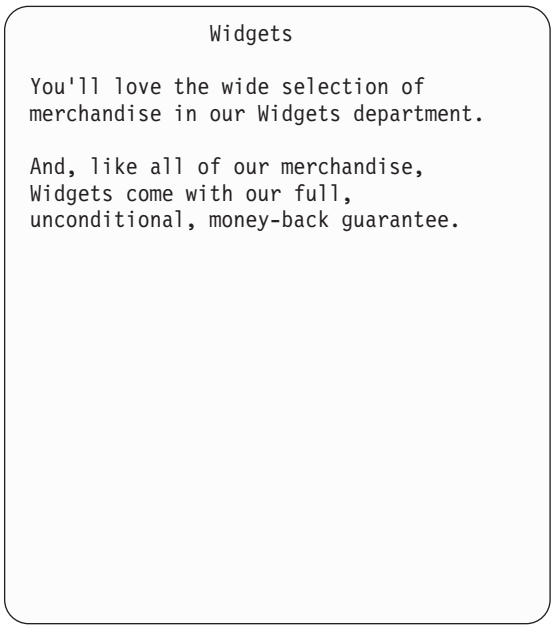


Figure 9. Entity Reference for Text Substitution

We can refer to the same entity in the text of the source file as many times as we like. And, if we should ever want to change the text of the entity, we only have to do it in one place–where we declared it in the declaration subset.

We’ll change the entity we declared earlier to show you what we mean.

```
<!doctype dm system [
<!entity guar "partial, conditional, non-refundable guarantee">
]>
<panel name=widget22 width=40>Widgets
  <area>
    <info width=38>
      <p>You'll love the wide selection of merchandise
      in our Widgets department.
      <p>And, like all of our merchandise, Widgets come
      with our &guar;.
    </info>
  </area>
</panel>
```

The only change we made was to the text of the entity declaration, not the entity name. Following reformatting, the text of the entity reference now looks like this:

```
                Widgets

You'll love the wide selection of
merchandise in our Widgets department.

And, like all of our merchandise,
Widgets come with our partial,
conditional, non-refundable guarantee.
```

Figure 10. Entity Reference for Text Substitution

If, for any reason you need to change the name of an entity, be sure to update all of the references to the entity name in your text.

You can also define the text of an entity in an external file and refer to that file in an entity declaration. If you do this, you must include the SYSTEM parameter in the entity declaration, to indicate to the conversion utility that the file is external.

Note: You must include the external file in the concatenation of DTL source files defined to the conversion utility.

For example, we'll define a text string we want to use as an entity in our source file in a file called WIDGETS. Here are the contents of the WIDGETS file:

```
doohickeys, whatnots, and gizmos
```

To declare this file in the entity declaration in our source file, we code it like this, with the SYSTEM parameter:

```
<!doctype dm system [
<!entity guar "full, unconditional, money-back guarantee">
<!entity widgets system>
]>
```

And, if we want to use the text string in our source file, we refer to the entity "widgets" (in this case, the file name also serves as the entity name).

```
<!doctype dm system [
<!entity guar "full, unconditional, money-back guarantee">
<!entity widgets system>
]>
<panel name=widget23 width=42>More Widgets
  <area>
    <info width=40>
      <p>The fine selection of items in our Widgets department
      includes &widgets;.
      <p>And, like all of our merchandise, Widgets come with
      our &guar;.
    </info>
  </area>
</panel>
```

Figure 11 shows the formatted result.

```
                More Widgets

  The fine selection of items in our
  Widgets department includes doohickeys,
  whatnots, and gizmos.

  And, like all of our merchandise,
  Widgets come with our full,
  unconditional, money-back guarantee.
```

Figure 11. Entity Reference for Text Substitution and File Embedding

Anytime we want to update or change the text of the entity, we only need to change the text in the WIDGETS file.

In the previous example, the name "widgets" serves as the external file name and as the entity name.

The SYSTEM parameter may optionally be followed by the filename for the included file. When the SYSTEM parameter is used but no filename is provided, the entity name is used as the filename.

For instance, if you want to declare a different entity name for the WIDGETS file, “things” for example, code it like this in the entity declaration:

```
<!doctype dm system [
<!entity guar "full, unconditional, money-back guarantee"
><!entity things system "widgets">
]>
```

And refer to the entity name, *things*, like this:

```
<!doctype dm system [
<!entity guar &"full, unconditional, money-back guarantee&">
<!entity things system "widgets">
]>
<panel name=widget24 width=42>More Widgets
  <area>
    <info width=40>
      <p>The fine selection of items in our Widgets department
      includes &things;.
      <p>And, like all of our merchandise, Widgets come with
      our &guar;.
    </info>
  </area>
</panel>
```

The formatted result of this markup is the same as that shown in Figure 11 on page 21, assuming no changes were made to the text of the WIDGETS file.

Parameter Entities

Parameter entities allow you to place multiple entity declarations within an external file and refer to them within a source file. To embed the entities into the source file, you must declare the external file as a parameter entity. A parameter entity is identified by a percent symbol (%) following the term “entity” and followed by a space and the entity name. See “Entity Declarations” on page 202 for the syntax description. You refer to a parameter entity within the DOCTYPE statement by preceding the entity name with a percent symbol (%) and following it with a semicolon (;). This embeds the parameter entity file and allows its entities to be referred to in the source file.

For example, we’ve declared all of our entities within an external file called SYMBOLS. Here are the contents of the SYMBOLS file:

```
<!ENTITY sb "ShelfBrowse">
<!ENTITY cotime "ten days">
<!ENTITY xcotime "five days">
<!ENTITY nttime "three days">
<!ENTITY nitem "red checkout card">
<!ENTITY lfine "ten cents">
<!ENTITY cophone "555-1234">
```

The conversion utility locates the parameter entity using the rules defined above for entity external files.

We can embed the above file into the declaration subset of the source file with a parameter entity declaration within the DOCTYPE statement. As long as we declare the parameter entity and refer to it properly, we can use any of the declared entities in the external file in the text of the source file.

```
<!doctype dm system
 [<!entity % SYMBOLS system> %SYMBOLS;]>
<panel name=chkout width=40 depth=22>Library Checkout Periods
  <area>
    <info width=38>
      <p>&sb; allows you to check out an inventory
```

```
      item for a maximum of &cotime;.
      However, you can renew the item for an additional
      &xcotime; by calling in your card number to our
      checkout phone line (&cophone;) any time of day.
      <p>If an inventory item is a new shelf item
      (indicated by the &nitem;), you may only reserve it for
      a maximum of &nttime;.
      You may not renew a new shelf item.
      <p>There is a fine of &lfine; per day for all
      items returned late.
    </info>
  </area>
</panel>
```

Figure 12 shows the formatted result.
Parameter entity names must have the following characteristics:

```
        Library Checkout Periods

  ShelfBrowse allows you to check out an
  inventory item for a maximum of ten
  days. However, you can renew the item
  for an additional five days by calling
  in your card number to our checkout
  phone line (555-1234) any time of day.

  If an inventory item is a new shelf
  item (indicated by the red checkout
  card), you may only reserve it for a
  maximum of three days. You may not
  renew a new shelf item.

  There is a fine of ten cents per day
  for all items returned late.
```

Figure 12. Parameter Entities

- 1–8 characters
- The first character must be alphabetic (A–Z, a–z, @, #, or $)
- Remaining characters, if any, must be A–Z, a–z, @, #, $, or 0–9
- Parameter entity names are case-sensitive.

Imbedding Source Files

You can also use entities to imbed entire files within your source file. For example, you could define common variables for several panels in your source file in a separate file. These separate files are stored as members of any input library specified to ISPDTLC. The following markup shows the contents of a file called VARDEFS.

```
<varclass name=titlcls type='char 50'>
<varclass name=bookcls type='char 20'>
<varclass name=pagecls type='char 5'>
<varclass name=datecls type='char 8'>

<varlist>
  <vardcl name=title    varclass=titlcls>
  <vardcl name=author   varclass=bookcls>
```

```
  <vardcl name=publish  varclass=bookcls>
  <vardcl name=pages    varclass=pagecls>
  <vardcl name=curdate  varclass=datecls>
</varlist>
```

Another common markup file could be defined for an action bar. The following markup shows a portion of the contents of a file called ACTNBAR.

```
<ab>
  <abc>File
    <pdc>Add Entry
      <action run=add>
    <pdc>Delete Entry
      <action run=delete>
    <pdc>Update Entry
      <action run=update>
    <pdc>Exit
      <action run=exit>
  <abc>View
.
.
.

  <abc>Options
.
.
.

  <abc>Help
.
.
.

</ab>
```

We can imbed these files in a source file by coding entity references to the files in the source file DOCTYPE statement.

```
<!doctype dm system [
<!entity actnbar system>
<!entity vardefs system>]>

&vardefs;

<panel name=dfdxmp21>Library Inventory

&actnbar;

  <topinst>To add a book to the inventory, complete the fields below,
           and then press Enter.
  <area>
    <dtafld datavar=title usage=in pmtwidth=14>Title
    <dtafld datavar=author usage=in pmtwidth=14>Author
    <dtafld datavar=publish pmtwidth=14>Publisher
    <dtafld datavar=pages usage=in pmtwidth=14>Number of pages
    <divider type=solid gutter=3>
    <dtafld datavar=curdate usage=out pmtwidth=20>Today's date is
  </area>
</panel>
```

The variable definitions in VARDEFS are referred to by the data fields in the panel because the file was imbedded into the source file through the entity declaration. In the previous example, the entry width information for each field is obtained from the variable definitions.

File imbed entity names must have the following characteristics:

- 1–8 characters
- The first character must be alphabetic (A–Z, a–z, @, #, or $)
- Remaining characters, if any, must be A–Z, a–z, @, #, $, or 0–9
- Entity names are case-sensitive.

Run-Time Substitution Variables

If you need to include a dialog variable within your panel source that will be substituted at run time, the output panel must be created to contain an "&variable" string. An example would be a reference to an ISPF variable such as &ZDATE.

The conversion utility always tries to substitute each "&variable" found at conversion time with the available entity definitions. If the conversion utility can't find an entity definition, it issues a warning message, and then passes the original "&variable" into the output panel.

To avoid the warning message, you can use the predefined entity "&amp". You can code the variable in the tag source as "&variable" to make "&variable" appear in the panel. Alternatively, you could provide an entity definition for the variable, such as <!ENTITY variable "&variable">.

You should use caution when designing panels that contain run-time substitution variables. The regular panel formatting process might not leave sufficient space in the panel text line for the variable value to be inserted. For example, a variable name of "&date" that requires 10 positions (YYYY/MM/DD) should be coded as "&date(10);".

Predefined Entities

The Dialog Tag Language provides you with a set of predefined entities that you can use in your source files. You can use them when the symbol you want is not present on your keyboard, or conflicts with a conversion utility delimiter symbol.

You do not need to declare a predefined entity to use it. If you use the entity in your source file as you would an entity that you declare within your document subset, the conversion utility performs the substitution for you. You should always use the pre-defined entities for all symbols that are used as part of the tag language syntax.

The Dialog Tag Language predefined entities include:

&gtsym;	greater than (>)
&ltsym;	less than (<)
:	colon (:)
&	ampersand (&)
;	semicolon (;)
.	period (.)
&quote;	single quote (')
&dquote;	double quote (")
–	short dash (–)
¬	not symbol (¬)
&us;	underscore (_)
∨	logical or (|)
&sll;	back slash (\)
&lbrk;	left bracket ([)
&rbrk;	right bracket (])
&lbrc;	left brace ({)
&rbrc;	right brace (})
−	minus sign (–)
+	plus sign (+)
&rbl;	required blank ( )
&tpl;	text placeholder ( )

&eqsym;	equal sign (=)
&rdb;	required SBCS blank in DBCS mode ( )
&percent;	percent sign (%)
˙	dot (.)
&cmdpmt	command prompt (= = = >)
&rptr	right pointer (-->)

Any of the predefined entities listed above can be coded with a replication factor. For example, &gtsym(5); will create the string '>>>>>' in the substituted text.

National Language text strings are also accessible as entities:

&more	More
&caution	CAUTION
&note	Note
&warning	Warning
&command	Command
&alpha	abcdefghijklmnopqrstuvwxyz
&scroll	Scroll
&option	Option
&horizdiv	|
&multihst	Enter "/" to select option
&multigui	Check box to select option
&release	Release
&maintlvl	Level:
&created	Created -
&datetext	Date:
&timetext	Time:
&notes	Notes
&attentn	Attention
&tutorial	Tutorial

Points to Remember:

1. Some of the symbols defined in the preceding list will not display on some non-programmable terminals.
2. The &rbl; predefined entity creates one blank in the resulting panel text. To place three required blanks in a text string, for example, you should code &rbl;&rbl;&rbl; in your tag source file.
3. The &tpl; predefined entity uses a hex FF code to reserve a space in DTL formatted text. After formatting is completed, the hex FF character is replaced by a blank. As with any predefined entity, you can change this default to another value. The current value of &tpl; is used for post-formatting text replacement. Thus, if you prefer to use an @ as the reserved space character, define the entity as follows:

   ```
   <! ENTITY TPL '@'>
   ```

   If multiple reserved spaces are required, you could use the following entity definitions to reserve 10 characters. To use your own entity name, first define TPL to override the system default character for text replacement. Second, add your entity definition, using the specified override character.

   ```
   <! ENTITY TPL '@'>
   <! ENTITY MYTPL '@@@@@@@@@@'>
   ```

   When the &tpl; is changed, be careful to select a character that will not otherwise be used in your panel.

4. The &rdb; predefined entity generates an SBCS blank when ISPDTLC is processing in DBCS mode, or a null character when processing in SBCS mode.
5. The ˙ predefined entity generates a dot (or period) character in the text. The number of spaces following the ˙ in the DTL source is maintained in the formatted panel.

Chapter 3. Getting Started: Designing Application Panels

Each application panel you create will serve a specific purpose, with unique fields, messages, and help information defined for each one. In this chapter, we tell you how to define elements that are common among application panels.This includes defining the application panels, and the interactive elements of panels, including action bars, instruction text, and command areas. We also tell you how to arrange the contents of your application panels using panel regions and dividers.

At the end of this chapter, we tell you about the PANDEF tag, which allows you to define common attributes and values for the panels in your application in a single place.

Defining Application Panels: The PANEL Tag

You use the PANEL tag, its associated attributes, and the required PANEL end tag to define an application panel and the specific characteristics of the panel.

The PANEL start and end tags define the beginning and ending of an application panel. The PANEL start tag defines:

- Panel name
- Name of the help panel for the application panel
- Name of the panel default
- Dimensions of the panel
- Associated key mapping list
- KEYLTYPE value
- APPLID value
- Cursor placement
- CCSID number
- MENU keyword
- PRIME keyword
- TUTOR keyword
- WINDOW value
- WINTITLE value
- APPTITLE value
- PAD value
- PADC value
- OUTLINE value
- EXPAND value.
- MSGLINE value
- TITLINE value
- CMDLINE value
- ATTRUSE value
- ENDATTR value
- TYPE value
- SMSG value
- LMSG value
- ASIS keyword
- ACTBAR keyword
- MERGESAREA value
- PANELSTMT value
- ENTKEYTEXT value
- IMAPNAME value

© Copyright IBM Corp. 1989, 2001

- IMAPROW value
- IMAPCOL value
- TMARGIN value
- BMARGIN value
- ERRORCHECK value
- ZUP value
- ZCONT value
- AUTONRET value
- AUTOTCMD value
- Panel title text

With the exception of the required NAME attribute used to identify the name of the application panel, all of the attributes for the PANEL tag are optional. Many attributes have default values that the conversion utility assumes if you do not specify the attribute. This section describes these attributes, and how to use them.

The PANEL start and end tags look like this, respectively:

```
<panel name=mainpan>
⋮

</panel>
```

In the preceding example, we included the required NAME attribute and its value *mainpan* on the PANEL start tag. ISPF requires that each panel definition contain this attribute and an associated value to identify the panel. The panel name is also used as the panel ID when the panel ID is displayed. The "NAME=*" notation will set the panel name to be the same as the member name of the input DTL source file. If multiple panel definitions have been combined within a single source file, then this notation should be used for only one panel definition within the file.

The *panel name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

Note: During conversion when the PREP option is active, the conversion utility uses a temporary PDS to store ISPF source format panels. The file name for interactive use is:

`'tsoprefix.TEMPDTLW.DTLPANnn', where nn is the screen number.`

Or, if the TSO NOPREFIX profile option is in effect, the file name is:

`'tsouserid.TEMPDTLW.DTLPANnn', where nn is the screen number.`

For batch, the file name is:

`'tsoprefix.TEMPDTLW.DTLBATCH.Ttttttt.Rnnnnn'`

Or, if the TSO NOPREFIX profile option is in effect, the file name is:

`'tsouserid.TEMPDTLW.DTLBATCH.Ttttttt.Rnnnnn'`

The batch file name is uniquely created for each ISPDTLC invocation by including the system time and a random number as the last two qualifiers of the name.

The ISPPREP utility is called to convert all of the generated panels from ISPF source format to pre-processed format at one time to improve performance.

The Panel Title

The text that appears as the title of the panel is called the title text. You define the title text by coding it as tag text for the PANEL start tag.

This example uses the text “Catalog Ordering System” as title text:

```
<panel name=mainpan>Catalog Ordering System
⋮

</panel>
```

Panel Size (Width and Depth)

Use the DEPTH and WIDTH attributes of the PANEL tag to define the size of an application panel. The PANEL tag has a default WIDTH value of 76 characters and a default DEPTH value of 22 lines. If you specify WINDOW=NO, the default WIDTH is 80 and the default DEPTH is 24. These are the values the conversion utility assumes if you do not specify dimensions for WIDTH and DEPTH.

The following example defines the panel size as 60 characters wide and 15 lines deep:

```
<panel name=mainpan width=60 depth=15>Catalog Ordering System
⋮

</panel>
```

If we want the width of the panel to be 76 characters wide (the default width), we only need to specify a value for DEPTH, as in this markup:

```
<panel name=mainpan depth=15>Catalog Ordering System
⋮

</panel>
```

This results in a panel with a default width of 76 characters and a specified depth of 15 lines.

Because you can display application panels in pop-ups, you should allow for pop-up borders (added by ISPF at run time) when you define the WIDTH and DEPTH values for application panels. When the panel is displayed in a pop-up, ISPF adds two lines to the depth specified and 4 characters to the width specified for pop-up borders. Remember that ISPF cannot display a panel whose size exceeds the device size and will issue an error message at run time if this situation is encountered.

Key Mapping Lists

To specify the function keys that are active for an application panel, use the KEYLIST attribute of the PANEL tag. This attribute specifies the name of the key mapping list you define for use with the panel. A key mapping list contains the keys that are active while the panel is displayed. The key mapping list also specifies what command is run when each key is pressed.

This PANEL definition refers to a key mapping list named *key01*:

```
<panel name=mainpan keylist=key01>Catalog Ordering System
⋮

</panel>
```

For more information on defining key mapping lists, see “Chapter 9. Defining Key Mapping Lists” on page 171.

Associated Help Panels

To provide help for an application panel (also called *extended help*), specify the name of the associated help panel with the HELP attribute of the PANEL tag. The help panel you specify appears when the user requests extended help while in the application panel or when contextual help is requested for an item on the panel, but no contextual help is available for the item. The help panel you specify is also displayed when the user requests extended help while in a contextual help panel associated with an item on the panel.

This panel definition refers to a help panel named *ordhelp*:

```
<panel name=mainpan help=ordhelp>Catalog Ordering System
⋮

</panel>
```

“Help Panels” on page 149 tells you how to create help panels for your application.

Panel Defaults

The PANEL tag attribute PANDEF provides the name of a panel default definition. Attribute values defined on the named PANDEF tag are used for the current panel unless the attribute has also been specified on the PANEL tag.

Cursor Placement

The PANEL tag attributes, CURSOR, CSRINDEX, and CSRPOS, allow you to specify where the cursor is placed when the panel is initially displayed. If you do not specify a specific cursor position, ISPF places the cursor in the first field in the PANEL definition that can contain the cursor.

Use the CURSOR attribute to specify the field that is to contain the cursor. Use the CSRINDEX and CSRPOS attributes to identify positions within the field you specify with the CURSOR attribute. CSRINDEX and CSRPOS can only be used when the CURSOR attribute is used.

The CURSOR Attribute

Use the CURSOR attribute to specify the value of the NAME attribute of a CHOICE or SELFLD tag, or the value of the DATAVAR attribute of a CHOFLD, DTAFLD or LSTCOL tag. The characteristics of cursor placement are described in the following list:

CHOFLD The cursor appears in the first character position of the choice field. Cursor positioning is valid only when the USAGE attribute of the CHOFLD tag specifies INPUT or BOTH.

CHOICE The cursor appears in the entry field of the specified choice in a multiple-choice selection field.

DTAFLD The cursor appears in the first character position of the data field. Cursor positioning is valid only when the USAGE attribute of the DTAFLD tag specifies INPUT or BOTH.

LSTCOL The cursor appears in the first row in the list column. Cursor positioning is valid only when the USAGE attribute of the LSTCOL tag specifies INPUT or BOTH.

SELFLD The cursor appears in the entry field of the specified single-choice selection field.

"Chapter 5. Application Panel Fields" on page 83 provides a complete description of the types of interactive fields you can define for your application panels.

You can also place the cursor in the command area of the panel by specifying *cmdarea* as the CURSOR value.

"Defining a Command Area" on page 55 provides a complete description of the CMDAREA tag.

In the following example[4], the CURSOR attribute specifies the data field DATAVAR value *place*. When the panel is initially displayed, the cursor appears in the first character position of that field. Figure 13 shows the formatted result.

```
<!doctype dm system>
<panel name=mainpan1 cursor=place>Travel Agency
    <selfld name=dest selwidth=50 pmtwidth=15>Destinations:
      <choice>London
      <choice>Madrid
      <choice>Paris
      <choice>Zurich
    </selfld>
    <divider>
    <dtafld datavar=place entwidth=9 pmtwidth=5>Other
<cmdarea>
</panel>
```

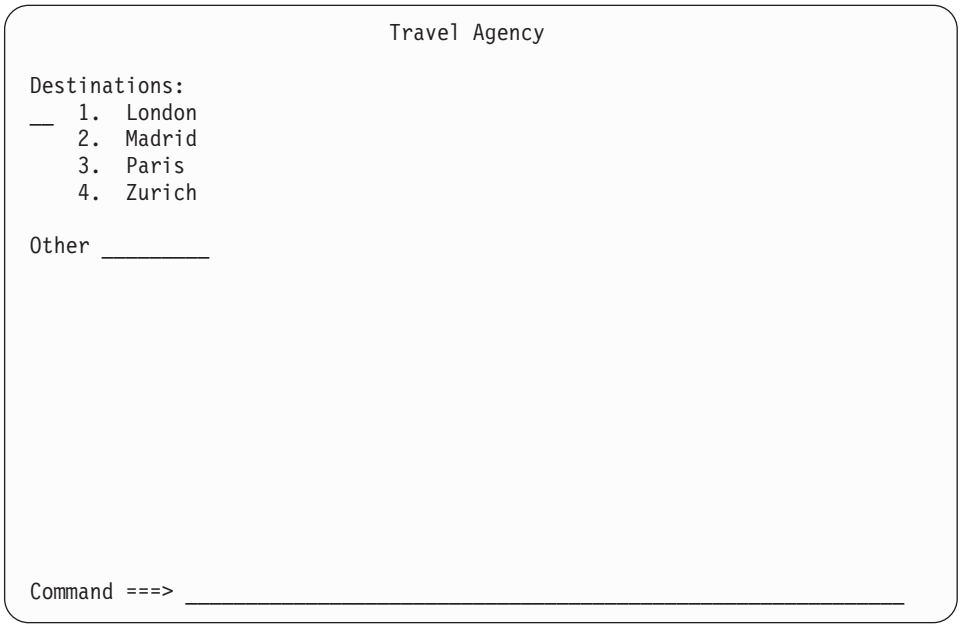


Figure 13. Cursor Placement

4. In this example, and in other examples in this chapter, we show tag markup for elements such as fields and variables that have not yet been discussed so that we can illustrate the formatting characteristics of some tags. The syntax of these elements are not important for the purposes of these examples. We discuss the syntax conventions of these elements in later chapters of this book.

If no cursor placement was specified in the PANEL tag for the preceding example, the cursor would appear in the entry field of the **Destinations** single-choice selection field when the panel is initially displayed.

The CSRINDEX Attribute

To place the cursor in a table row within a list field, use the CURSOR attribute to specify the data variable name for a list column within the list field, and the CSRINDEX attribute to specify the table row number where the cursor should be placed. The value you assign to CSRINDEX must be numeric.

The CSRPOS Attribute

If you use the CURSOR attribute to place the cursor within an input-only, or input/output data field or list column, or the command area, you can also define a specific character position for the cursor using the CSRPOS attribute.

The value you assign to the CSRPOS attribute must be numeric. This numeric value indicates the number of character positions from the left margin of the field where the cursor is placed, where a 1 specifies that the cursor should be in the first character position.

Other Panel Attributes

See "PANEL (Panel)" on page 415 for more information.

KEYLTYPE
This attribute is used to add the SHARED keyword to the KEYLIST parameter of the)PANEL statement.

APPLID
This attribute is used to add the application ID to the KEYLIST parameter of the)PANEL statement.

CCSID
This attribute specifies the coded-character-set identifier as defined by the Character Data Representation Architecture.

MENU
This attribute specifies that the panel will be an ISPF menu selection panel.

PRIME
This attribute is used with the MENU attribute to specify an ISPF primary option menu.

TUTOR
This attribute specifies that the panel is to be an ISPF tutorial panel.

WINDOW
This attribute is used to control the generation of the WINDOW keyword on the panel)BODY statement.

WINTITLE
This attribute is used to add a title on a pop-up window border.

APPTITLE
This attribute is used to add a title on the GUI window border.

PAD
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

PADC
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

OUTLINE
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

EXPAND
This attribute causes the ISPF EXPAND keyword to be added to the panel)BODY statement.

MSGLINE
This attribute controls the provision for a long message line in the generated panel. When MSGLINE=NO, the blank line for the long message is not added to the panel)BODY section.

TITLINE
This attribute controls the provision for a panel title line in the generated panel. When TITLINE=NO, the title line is not added to the panel)BODY section. This attribute allows a panel formatted as a dynamic area to provide the panel title as part of the dynamic area data.

CMDLINE
This attribute controls the automatic addition of the command area to a menu selection or table display panel. When CMDLINE=NO, the command area is not automatically generated when the CMDAREA tag is not present in the DTL source file.

ATTRUSE
This attribute controls the use of panel attribute characters in the range of x'01' through x'2F'. When ATTRUSE=YES, dynamic area attributes (specified with the ATTR tag) can be assigned low-order hex values normally reserved for use by the conversion utility.

ENDATTR
This attribute specifies that when the last attribute on any panel body line is 'normal text' (CUA), it will be replaced by the default 'text' (ISPF) attribute.

TYPE
This attribute specifies that the panel will be used for host display, GUI mode display, or both.

SMSG
This attribute provides the name of the field where the short message is to be placed.

LMSG
This attribute provides the name of the field where the long message is to be placed.

ASIS
This attribute specifies that the command and long message fields are to appear on the display as specified in the generated panel definition. When ASIS is specified, any user request specified on the Settings panel, or by setting the system variable ZPLACE is ignored.

ACTBAR
This attribute causes the action bar information for the panel to be generated, overriding the NOACTBAR invocation option.

MERGESAREA
This attribute specifies that a panel with a single scrollable area be reformatted to combine the scrollable area into the panel body.

PANELSTMT
This attribute controls the creation of the)PANEL statement.

ENTKEYTEXT
This attribute provides the text for the Enter key push button provided on panels displayed in GUI mode.

IMAPNAME
This attribute provides the name of the image placed on panels displayed in GUI mode.

IMAPROW
This attribute provides the row number for positioning the image.

IMAPCOL
This attribute provides the column number for positioning the image.

TMARGIN
This attribute provides the number of blank lines to format at the top of the panel as a top margin.

BMARGIN
This attribute provides the number of blank lines to format at the bottom of the panel as a bottom margin.

ERRORCHECK
This attribute specifies that error checking code is added to the)PROC panel section.

ZUP
This attribute provides the name of the tutorial panel to be assigned to the ZUP variable.

ZCONT
This attribute provides the name of the tutorial panel to be assigned to the ZCONT variable.

Defining Action Bars and Pull-Downs

To create a consistent user interface, you should design your applications according to the object-action process sequence defined by the SAA Common User Access. The action bar is a major user interface component that helps you achieve consistency in your applications.

The action bar is the panel element located at the top of an application panel that contains action bar choices for the panel. Each action bar choice represents a group of related choices that appear in the pull-down associated with the action bar choice. When the user selects an action bar choice, the associated pull-down appears directly below the action bar choice. Pull-downs contain choices that, when selected by the user, perform actions that apply to the contents of the panel.

Panel Design Note

ISPF and DTL provide the tools to help you create the object-action process sequence in your application, but it is your responsibility as an application designer to ensure that the contents of your action bar are actions that can be applied to the objects contained within your panel.

Typically, application panels intended for display within primary windows contain action bars that present the user with all of the available actions that apply to that panel. Application panels that are displayed as pop-ups should not include the action bar. Instead, actions for a pop-up panel are presented in the function key area.

Figure 14 shows an action bar with the **File** pull-down menu displayed.

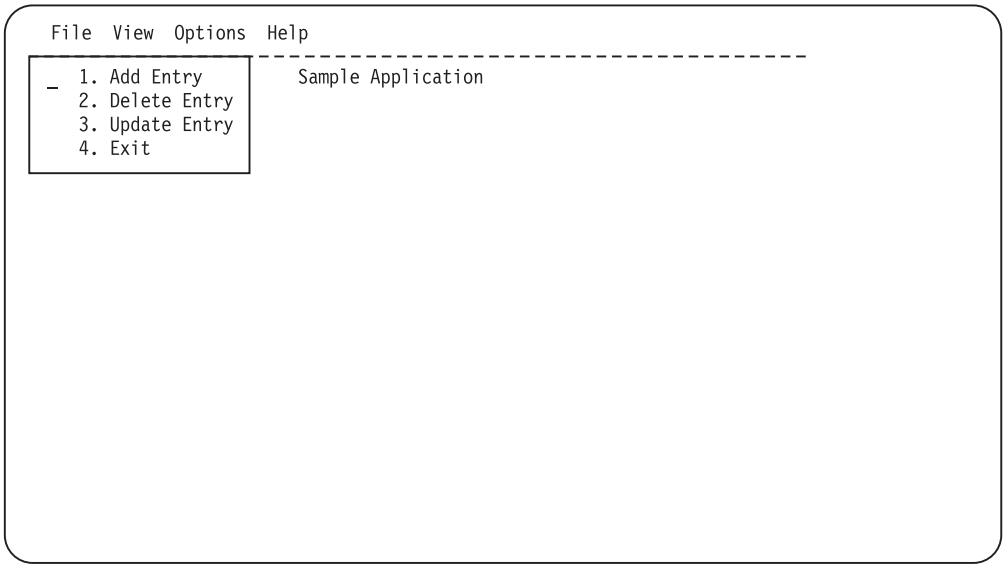


Figure 14. Action Bar and Pull-Down

The tags you use to create the action bar and its pull-down menus are:

AB To start an action bar definition. The required AB end tag ends an action bar definition.

ABC To define each of the action bar choices.

PDC To define the choices on the pull-down associated with an action bar choice.

ACTION To specify an action to be taken when the pull-down choice is selected. The ACTION tag is coded within the PDC tag.

M To specify a mnemonic character for action bar choice or pull-down choice selection.

Coding an Action Bar Definition

The following list describes how to code an action bar definition:

- Code the AB start tag immediately after the PANEL start tag and before any other tags in the panel.
- Following the AB start tag, code an ABC tag for each action bar choice in the action bar. The text you specify on the ABC tag is the text that appears in the action bar as the action bar choice.
- Code the associated PDC tags within the ABC tags. The text you specify on the PDC tag is the text that appears as the pull-down choice.
- Following each PDC tag, code one or more ACTION tags to specify what type of action occurs when that pull-down choice is selected by the user.

  The ACTION tag RUN attribute (and its *internal-command-name*) define a command action for the pull-down choice.If you define multiple ACTION tags for a pull-down choice, one of which contains a RUN value, code the RUN action last, because any actions specified after a RUN action are ignored.
- End the action bar definition with the required AB end tag.

The following example shows the markup for the action bar shown in Figure 14 on page 37. The detailed markup for the **File** pull-down is included.

```
<panel name=mainpan2 depth=15>Sample Application
  <ab>
    <abc>File
      <pdc>Add Entry
        <action run=add>
      <pdc>Delete Entry
        <action run=delete>
      <pdc>Update Entry
        <action run=update>
      <pdc>Exit
        <action run=Exit>
    <abc>View
.
.
.
    <abc>Options
.
.
.
    <abc>Help
.
.
.
  </ab>
.
.
.
</panel>
```

Pull-Down Choice Actions

A pull-down choice provides an immediate action to the user. To ensure that a pull-down choice performs an immediate action, you should code an ACTION tag that specifies the RUN attribute for each pull-down choice. The value you assign to RUN tells ISPF which command to run when the user selects the choice.

In the preceding markup example, each ACTION definition uses the RUN attribute to specify a command. Each of these commands must be defined within the command table for the application. “Chapter 8. The Application Command Table” on page 165 tells you how to define commands for an application.

In addition to the RUN action, you can specify other types of actions to occur when a pull-down choice is selected. The SETVAR and TOGVAR attributes on the ACTION tag can be used to set and toggle variables which the application can use to determine the processing to perform.

Remember, any SETVAR or TOGVAR actions for a pull-down choice must be coded before any ACTION definition specifying the RUN action, because actions coded after RUN are ignored.

A pull-down choice may be marked as unavailable. The UNAVAIL attribute is used to provide a variable name that is used by ISPF to determine the availability of the pull-down choice. When the variable value is 1, the pull-down choice is unavailable.

Action Bar Help

You can provide help for each action bar choice and pull-down choice with the HELP attribute on the ABC and PDC tags, respectively. By specifying the name of a help panel or message for the action bar choice or pull-down choice, ISPF knows which help information to display when the user requests help on that choice. If you do not specify help for a pull-down choice, the help for the action bar choice is displayed, when the user requests help. If there is no help defined for the action bar choice, the extended help panel is displayed.

In the following example, we've added the HELP attribute to each of the action bar choices and pull-down choices in the action bar defined on page 38. The values specified with each HELP attribute are the NAME values of defined help panels.

```
<!doctype dm system>
<panel name=mainpan3 width=50 depth=15>Sample Application
  <ab>
    <abc help=hfile>File
      <pdc help=hnew>Add Entry
        <action run=add>
      <pdc help=hopen>Delete Entry
        <action run=delete>
      <pdc help=hsave>Update Entry
        <action run=update>
      <pdc help=hexit>Exit
        <action run=Exit>
    <abc help=hview>View
⋮

    <abc help=hoption>Options
⋮

    <abc help=hhelp>Help
⋮

  </ab>
⋮

</panel>
```

In the preceding example, we defined a help panel named *hhelp* for the **Help** action bar choice.

Common User Access requires that you put the **Help** action bar choice as the last action bar choice in an action bar definition. You should code the **Help** action bar pull-down as follows:

```
<abc help=hhelp>Help
  <pdc>Extended help
    <action run=exhelp>
  <pdc>Keys help
    <action run=keyshelp>
</abc>
```

"Help Panels" on page 149 tells you how to define help panels.

Preselected Pull-Down Choices

You can define a pull-down choice as being *preselected* with the CHECKVAR and MATCH attributes of the PDC tag.The CHECKVAR attribute specifies the name of a variable that you set at run time to indicate if the pull-down choice should be preselected. The MATCH attribute defines a value that causes the choice to be preselected. ISPF compares the value of the variable named for the CHECKVAR attribute to the MATCH value, and if they are equal, the choice appears preselected when the pull-down is displayed.

Continuing with the library application, assume that the user can view the files in the library sorted by name, owner, date, or size. Preselecting a pull-down choice provides a visual cue to the user of the current sort order.

To preselect any of the pull-down choices, the same CHECKVAR value is specified for each choice, and a unique MATCH value is specified for each choice. The application variable specified with CHECKVAR is set to the MATCH value to indicate the sorting option being used. The variable specified with CHECKVAR is changed each time the sorting option is changed. This provides a visual reminder to the user of how the files are sorted.

Mnemonic Choice Selection

ISPF supports mnemonic selection of action bar choices for both host system and GUI mode display and pull-down choices when running in GUI mode.

Mnemonic selection of action bar choices and pull-down choices is automatically determined by ISPDTLC when a non-DBCS conversion is in process. When DBCS is specified, mnemonics are not automatically generated. The default mnemonic character generation can be overridden by adding the MNEMGEN=NO attribute to the AB tag for non-DBCS conversions. The mnemonic character that will be selected is the first alphabetic or numeric character from the current action bar choice or pull-down choice description text that is not previously used as a mnemonic character within the action bar or current pull-down. If a unique mnemonic character cannot be selected, the conversion utility will issue a message. DBCS characters cannot be specified as mnemonics. See "M (Mnemonic)" on page 392 for a description of how to provide a mnemonic character that is not part of the normal choice description.

Mnemonic selection of action bar choices and pull-down choices may be specified by placing the M tag immediately in front of the character to be used as a mnemonic within the ABC or PDC text.

The automatic mnemonic generation does not replace any valid mnemonic specified by the M tag. (If the mnemonic character specified by the M tag is a duplicate of a mnemonic character previously selected by the generation process, a message is issued and ISPDTLC will attempt to replace the duplicate value that was specified.) This processing allows the combination of specific character selection with the automatic generation feature, as long as the characters automatically generated and the characters specified (by the M tag) are unique.

```
<:doctype dm system (
  <:ENTITY actnfile system>
  <:ENTITY actnoptn system>
  <:ENTITY actnhelp system>
)>
```

```
<panel name=pdcxmp1>Sample Application
  <ab>
    &actnfile;

    <abc>View
      <pdc checkvar=sorttype match=N><M>Name
        <action run=name>
      <pdc checkvar=sorttype match=O><M>Owner
        <action run=owner>
      <pdc checkvar=sorttype match=D><M>Date
        <action run=date>
      <pdc checkvar=sorttype match=S><M>Size
        <action run=size>

    &actnoptn;
    &actnhelp;
  </ab>

  <topinst>
  <area>
  </area>
  <cmdarea>
</panel>
```

If the application sets the variable *sorttype* to "D" before the panel is displayed, then the **Date** choice will be preselected.

Figure 15 shows how the **View** pull-down would appear in this scenario.

Pull-Down Choice Accelerator Support

ISPF supports accelerators on pull-down choices when you are operating in GUI mode. Accelerators may include up to three keys. They are supported in DTL by specifying the ACC1, ACC2, and ACC3 attributes on the PDC tag.

See "PDC (Pull-Down Choice)" on page 430 for a description of accelerator attributes.

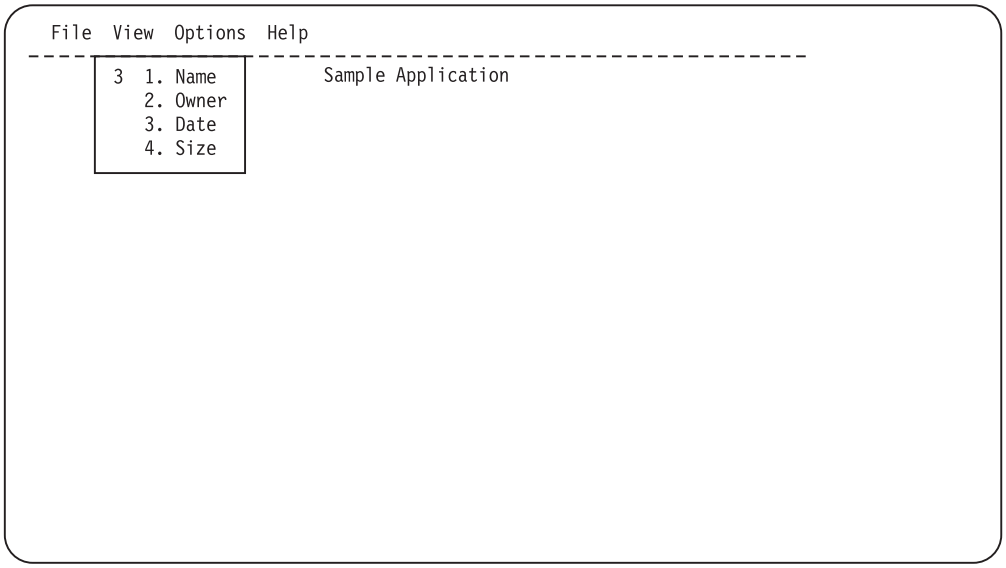


Figure 15. Preselected Pull-Down Choice

Defining the Panel Body

In this section, we tell you how to use DTL to define elements of the panel body such as instruction text, areas, regions, and dividers.

Panel Instructions

DTL provides you with the TOPINST, PNLINST, and BOTINST tags to define instructions for your application panels. None of the tags have required end tags associated with them.

Use the instruction tags to provide text that tells the user how to interact with the panel or how to continue with an application.

If the COMPACT attribute is not specified, a blank line is added to the panel after each TOPINST tag and before each PNLINST or BOTINST tag.

You must code the top and bottom instruction tags outside of the portion of the panel defined with the AREA tag and its matching end tag. (The section called "The AREA Tag" on page 43 explains how to use the AREA tag). Code the TOPINST tag immediately after the action bar definition (or the PANEL start tag if the panel does not contain an action bar). Code the BOTINST following the main body of the panel, before the PANEL end tag. You may code PNLINST tags within the AREA tag.

This application panel markup contains both types of instructions. Figure 16 on page 43 shows the results.

```
<!doctype dm system>
<panel name=mainpan5>Item Selection
  <topinst>Select one of the following items and press Enter.
    <selfld name=itemtyp selwidth=76>
      <choice>Automotive
      <choice>Hardware
      <choice>Health and beauty
      <choice>Lawn and garden
      <choice>Sporting goods
    </selfld>
  <botinst>To exit the application, press F3.
</panel>
```

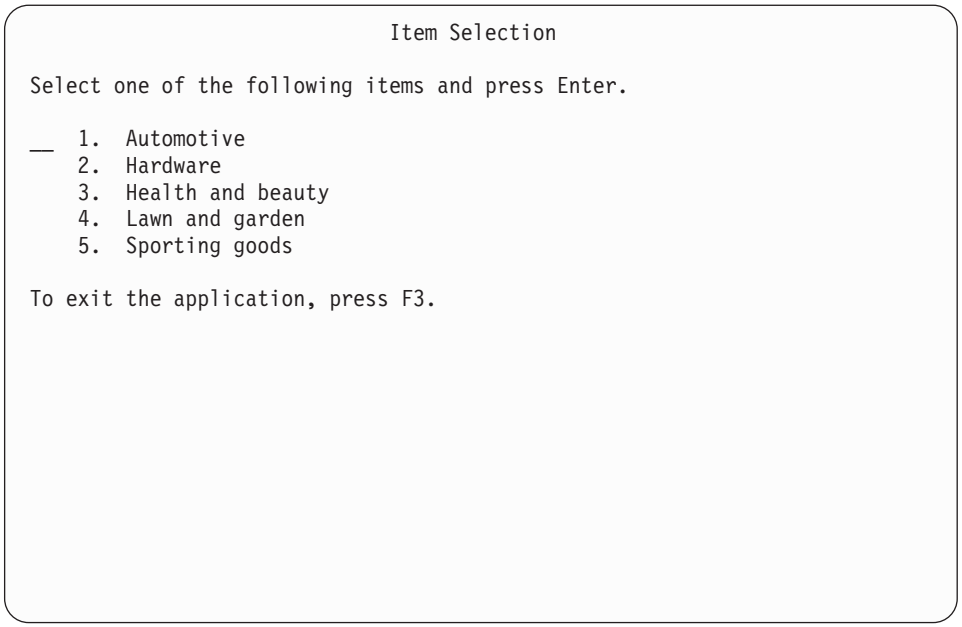


Figure 16. Top and Bottom Instructions

The AREA Tag

The AREA tag (and its matching end tag) defines the main portions of the panel body. You code all of the interactive fields for the panel within AREA definitions.

Add an AREA definition to the previous application panel markup.

```
<!doctype dm system>
<panel name=mainpan6 depth=18>Item Selection
  <topinst>Select one of the following items and press Enter.
  <area>
    <selfld name=itemtyp selwidth=76>
      <choice>Automotive
      <choice>Hardware
      <choice>Health and beauty
      <choice>Lawn and garden
      <choice>Sporting goods
    </selfld>
  </area>
  <botinst>To exit the application, press F3.
</panel>
```

As stated on page 42, you must code the top and bottom instruction tags outside of the AREA definition. In this example, we coded only a selection field within the AREA definition.

The AREA tag has an optional MARGINW attribute that you can use to specify the width of the panel body margins. This is useful for arranging fields on a panel.

The MARGINW attribute has a default value of *1*. You can specify a different value to increase the size of the panel body margins. For example, we could specify a margin width for the AREA in the preceding markup example.

```
<!doctype dm system>
<panel name=mainpan7>Item Selection
  <topinst>Select one of the following items and press Enter.
  <area marginw=10>
    <selfld name=itemtyp selwidth=58>
      <choice>Automotive
      <choice>Hardware
      <choice>Health and beauty
      <choice>Lawn and garden
      <choice>Sporting goods
    </selfld>
  </area>
  <botinst>To exit the application, press F3.
</panel>
```

We specified a margin width of *10*. Here's how the panel looks now:

```
                              Item Selection

 Select one of the following items and press Enter.

          __  1.  Automotive
              2.  Hardware
              3.  Health and beauty
              4.  Lawn and garden
              5.  Sporting goods

 To exit the application, press F3.
```

Figure 17. AREA MARGINW=10

Scrollable Areas

You specify a scrollable area with the Dialog Tag Language by coding the AREA tag and specifying the DEPTH, EXTEND, and DIV attributes for the area. When the DEPTH attribute is present, the conversion utility generates the)AREA section in the panel definition, along with the corresponding)ATTR and)BODY entries for the scrollable area.

Help panels generated by the Conversion Utility that contain all of the help panel text within an AREA tag (with DEPTH specified) are not split into separate panels. The conversion utility places the text in an)AREA section, which allows you to define panels up to the display size limit of ISPF.

If you specified DEPTH to signal the creation of a panel with a scrollable area, you can also specify the EXTEND and DIV attributes.

You can specify EXTEND=ON to allow the panel to expand to the logical window size. If you intend to have the panel in a pop-up window, you should not code the EXTEND attribute. Panels that specify EXTEND=ON cannot be preprocessed.

You use the DIV attribute to control the creation of a divider line before and after the scrollable area.

If you specify DIV=BLANK, a blank divider line is added before and after the area. If you specify DIV=SOLID, a visible divider is created. The visible divider formats with an attribute byte on each end of the line of dashes, which causes the line to appear with a 1 character "space" on both ends. Omitting the DIV attribute or specifying DIV=NONE causes the area to be created without divider lines.

The conversion utility uses the DEPTH attribute value to reserve a fixed amount of space in the panel body. This space, together with the divider lines, if specified, is considered as part of the body within the depth limit specified (or defaulted) on the PANEL tag. When EXTEND=OFF, the minimum depth for a scrollable area is two lines, one for the scrolling indicator line and at least one line of displayable text.

The following markup illustrates how you would code a scrollable panel. Figure 18 on page 46, Figure 19 on page 46, and Figure 20 on page 47 show the formatted result.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar2 system>
  <!entity sampabc system>)>
&sampvar2;

<PANEL NAME=scrarea3 KEYLIST=keylxmp>File-A-Case
<AB>
&sampabc;
</AB>
<TOPINST COMPACT>
         Type in client's name and case number (if applicable).
<TOPINST>Then select an action bar choice.
<AREA>
<DTAFLD DATAVAR=caseno PMTWIDTH=12 ENTWIDTH=7 DESWIDTH=25>Case No
   <DTAFLDD>(A 7-digit number)
<DTAFLD DATAVAR=name PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25>Name
   <DTAFLDD>(Last, First, M.I.)
<DTAFLD DATAVAR=address PMTWIDTH=12 ENTWIDTH=25>Address
<DIVIDER>
<SELFLD NAME=casesel PMTWIDTH=30 PMTLOC=before SELWIDTH=38>Choose
one of the following
   <CHOICE CHECKVAR=case MATCH=civ>Civil
   <CHOICE CHECKVAR=case MATCH=estate>Real Estate
   <CHOICE CHECKVAR=case MATCH=environ>Environmental
</SELFLD>
</AREA>
<AREA DEPTH=6>
<SELFLD TYPE=multi PMTWIDTH=35 SELWIDTH=50>Check type of offense committed
   <CHOICE NAME=patin HELP=patin CHECKVAR=val>Patent Infringement
   <CHOICE NAME=defa HELP=defame CHECKVAR=def>Defamation
   <CHOICE NAME=cont HELP=cont CHECKVAR=con>Breach of Valid Contract
   <CHOICE NAME=priv HELP=priv CHECKVAR=pri>Invasion of Privacy
   <CHOICE NAME=incr HELP=incr CHECKVAR=icr>Interference with Contractual
            Relations
   <CHOICE NAME=disp HELP=disp CHECKVAR=dis>Improper Disposal of Medical
            By-Products
   <CHOICE NAME=fraud HELP=fraud CHECKVAR=fra>Fraud
```

```
</SELFLD>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

```
  File  Search  Help
------------------------------------------------------------------------
                              File-A-Case

Type in client's name and case number (if applicable).
Then select an action bar choice.

Case No  . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following    __  1.  Civil
                                   2.  Real Estate
                                   3.  Environmental
                                                        More:     +
Check type of offense committed
_  Patent Infringement
_  Defamation
_  Breach of Valid Contract
Enter a command ===> ___________________________________________________
 F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

Figure 18. Scrollable Panel Area

After scrolling, the panel appears as follows:

```
  File  Search  Help
------------------------------------------------------------------------
                              File-A-Case

Type in client's name and case number (if applicable).
Then select an action bar choice.

Case No  . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following    __  1.  Civil
                                   2.  Real Estate
                                   3.  Environmental
                                                        More:   - +
_  Breach of Valid Contract
_  Invasion of Privacy
_  Interference with Contractual Relations
_  Improper Disposal of Medical By-products
Enter a command ===> ___________________________________________________
 F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

Figure 19. Application Panel Area

After scrolling, the last choice in the list is visible.

```
  File  Search  Help
------------------------------------------------------------------------
                              File-A-Case

Type in client's name and case number (if applicable).
Then select an action bar choice.

Case No  . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following    __  1.  Civil
                                   2.  Real Estate
                                   3.  Environmental
                                                          More:   -
_  Invasion of Privacy
_  Interference with Contractual Relations
_  Improper Disposal of Medical By-products
_  Fraud
_
Enter a command ===> ____________________________________________________
 F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

Figure 20. Scrollable Panel Area

Multiple AREA Tags

The default AREA tag formatting arranges areas vertically within the panel body.

The WIDTH and DIR attributes of the AREA tag allow areas to be formatted horizontally.

The following markup illustrates the use of horizontal areas. Figure 21 on page 49 shows the formatted result.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar2 system>
  <!entity sampabc system>)>
&sampvar2;

<PANEL NAME=scrarea4 KEYLIST=keylxmp>File-A-Case
<AB>
&sampabc;
</AB>
<CMDAREA>Enter a command
<TOPINST COMPACT>
         Type in client's name and case number (if applicable).
<TOPINST>Then select an action bar choice.
<AREA width=50 dir=horiz>
<DTAFLD DATAVAR=caseno PMTWIDTH=12 ENTWIDTH=7 DESWIDTH=21>Case No
   <DTAFLDD>(A 7-digit number)
<DTAFLD DATAVAR=name PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=8>Name
   <DTAFLDD>(Last, First, M.I.)
<DTAFLD DATAVAR=address PMTWIDTH=12 ENTWIDTH=25>Address
<DIVIDER>
<SELFLD NAME=casesel PMTWIDTH=11 PMTLOC=before SELWIDTH=38>Choose
one of the following
   <CHOICE CHECKVAR=case MATCH=civ>Civil
   <CHOICE CHECKVAR=case MATCH=estate>Real Estate
   <CHOICE CHECKVAR=case MATCH=environ>Environmental
</SELFLD>
```

```
</AREA>
<AREA width=26 dir=horiz>
<SELFLD TYPE=multi PMTWIDTH=24 SELWIDTH=26 depth=10>
  Check type of offense
   <CHOICE NAME=patin HELP=patin CHECKVAR=val>Patent Infringement
   <CHOICE NAME=defa HELP=defame CHECKVAR=def>Defamation
   <CHOICE NAME=cont HELP=cont CHECKVAR=con>Breach of Valid Contract
   <CHOICE NAME=priv HELP=priv CHECKVAR=pri>Invasion of Privacy
   <CHOICE NAME=incr HELP=incr CHECKVAR=icr>Interference with Contractual
           Relations
   <CHOICE NAME=disp HELP=disp CHECKVAR=dis>Improper Disposal of Medical
           By-Products
   <CHOICE NAME=fraud HELP=fraud CHECKVAR=fra>Fraud
</SELFLD>
</AREA>
</PANEL>
```

```
  File  Search  Help
------------------------------------------------------------------------
                              File-A-Case

Type in client's name and case number (if applicable).
Then select an action bar choice.
                                          Check type of offense
                                          #SAREA37                #
Case No  . . _______  (A 7-digit number)  #                       #
Name . . . . _____________________ (Last,  #                       #
                                   First,  #                       #
                                   M.I.)   #                       #
Address  . . _____________________         #                       #
                                          #                       #
Choose one                                #                       #
of the                                    #                       #
following   __  1.  Civil                 #                       #
                2.  Real Estate
                3.  Environmental
Enter a command ===> ___________________________________________________
 F1=Help       F3=Exit       F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

The contents of the scrollable area are as follows:

```
    )AREA SAREA37
```

```
_  Patent Infringement
_  Defamation
_  Breach of Valid
   Contract
_  Invasion of Privacy
_  Interference with
   Contractual Relations
_  Improper Disposal of
   Medical By-Products
_  Fraud
```

```
    )AREA SAREA37
```

Figure 21. Multiple Horizontal Areas

The DYNAMIC AREA Tag

You specify a dynamic area in the)BODY section by coding the DA and ATTR tags. The DA tag is used to define the dynamic area in the panel)BODY section. The ATTR tag is used to specify the)ATTR section entries for DATAIN, DATAOUT, and CHAR attribute types used within the dynamic area. A dynamic area allows you to specify an area of the panel to format with your application. Refer to the *ISPF Dialog Developer's Guide and Reference* for more information.

The GRAPHIC AREA Tag

You specify a graphic area in the panel)BODY section by coding the GA tag. A Graphic area allows you to define a specific portion of the screen for a GDDM display. Refer to the *ISPF Dialog Developer's Guide and Reference* for more information.

The DIVIDER Tag

You can separate the elements on a panel or the regions you define for a panel with the DIVIDER tag. A DIVIDER definition produces either a blank or visible divider line, depending on the value you assign to the TYPE attribute of the DIVIDER tag. The visible divider line can be a dashed line or a solid line, or it can contain text.

The default value, NONE, produces a blank divider line. The values DASH, SOLID, and TEXT produce a visible divider line.

For horizontally formatted dividers,

- When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line.
- When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line.

Both SOLID and DASH use the "|" character (obtained from the ISPF literals table) for vertically formatted dividers. The value TEXT (used in combination with the FORMAT attribute) is valid only for dividers within vertical regions and specifies that blank padding is used for the supplied text.

The GAP attribute specifies whether the divider line completely crosses the panel area or region that contains the divider, or if a 1-character gap is to remain at either end of a horizontally formatted divider. The valid values for the GAP attribute are YES (the default), and NO.

The value you assign to GUTTER specifies the size (in characters) of the total width of the divider. For vertical formatting the default is 1, because ISPF allots 1 line of screen space for the divider. For horizontal formatting the default GUTTER size is 3, because an attribute byte is placed both before and after the divider character. Any value above the default is split to either side of the divider. If the GUTTER value is an even number, the conversion utility increases the number by 1 so that the divider is centered within the defined width. The GUTTER attribute is useful for creating blank space on a panel.

The NOENDATTR attribute is valid only when formatting dividers within horizontal regions. When NOENDATTR is specified, the ending attribute is not added to the divider. With NOENDATTR and a GUTTER size of 1, a divider of one blank character can be created. With a GUTTER size of 2, TYPE=SOLID can be used to produce a visible divider.

The FORMAT attribute is valid only when formatting dividers within vertical regions. The FORMAT attribute must be specified to have ISPDTLC process the text provided with the DIVIDER tag. FORMAT specifies the text placement within the divider line as START, CENTER, or END.

There are two DIVIDER tags defined in the following example. The first DIVIDER does not specify a TYPE attribute, and produces a blank horizontal line. The second DIVIDER specifies TYPE=SOLID, and produces a visible divider.

```
<!doctype dm system>
<panel name=fields1>Selections
  <area>
    <dtacol selwidth=24 pmtwidth=15>
    <selfld name=item>Pick an item:
      <choice>Widget
      <choice>Doohickey
      <choice>Gizmo
    </selfld>
    <divider>
    <selfld name=color>Pick a color:
      <choice>Red
      <choice>Green
    </selfld>
    <divider type=solid gap=no>
    <selfld name=size>Pick a size:
      <choice>Miniscule
      <choice>Behemoth
    </selfld>
    </dtacol>
  </area>
  <botinst>To exit the application, press F3.
</panel>
```

Figure 22 shows the result:

```
                                Selections

Pick an item:
__  1.  Widget
    2.  Doohickey
    3.  Gizmo

Pick a color:
__  1.  Red
    2.  Green
--------------------------------------------------------------------------------
Pick a size:

__  1.  Miniscule
    2.  Behemoth

To exit the application, press F3.
```

Figure 22. Area Dividers

The dashed line in the second divider in the preceding example extends across the entire AREA definition to both margins because we specified GAP=NO in the DIVIDER definition.

The REGION Tag

You can further define the areas of your panel, and how you want the information in the areas arranged, with the REGION tag. Using one or more regions within a PANEL definition provides an easy way of arranging the elements on a panel. Like the PANEL and AREA tags, the REGION end tag is required.

The DIR (direction) attribute of the REGION tag specifies how the elements within a region are arranged, either horizontally or vertically. The default value is VERT, which arranges the elements within the region vertically. This means that if you do not specify a horizontal region (DIR=HORIZ), or if you do not define a region at all, the panel elements are arranged vertically by default.

For example, the selection fields in the following example are arranged vertically, because no DIR value is defined for the REGION tag.

```
<!doctype dm system>
<panel name=fields2>Selections
  <area>
    <region>
      <dtacol selwidth=24 pmtwidth=15>
      <selfld name=item>Pick an item:
        <choice>Widget
        <choice>Doohickey
        <choice>Gizmo
      </selfld>
      <divider>
      <selfld name=color>Pick a color:
        <choice>Red
        <choice>Green
      </selfld>
      <divider type=solid gap=no>
      <selfld name=size>Pick a size:
        <choice>Miniscule
        <choice>Behemoth
      </selfld>
      </dtacol>
    </region>
  </area>
  <botinst>To exit the application, press F3.
</panel>
```

```
                                Selections

 Pick an item:
 __  1.  Widget
     2.  Doohickey
     3.  Gizmo

 Pick a color:
 __  1.  Red
     2.  Green
 ------------------------------------------------------------------------------

 Pick a size:
 __  1.  Miniscule
     2.  Behemoth

 To exit the application, press F3.
```

Figure 23. Vertical Region

We'll specify the HORIZ value for the region to change the layout of the selection fields to horizontal. Figure 24 shows the result.

```
<!doctype dm system>
<panel name=fields3>Selections
  <area>
    <region dir=horiz>
      <dtacol selwidth=20 pmtwidth=15>
      <selfld name=item>Pick an item:
        <choice>Widget
        <choice>Doohickey
        <choice>Gizmo
      </selfld>
      <divider type=solid gutter=5>
      <selfld name=color>Pick a color:
        <choice>Red
        <choice>Green
      </selfld>
      <divider type=solid gutter=5>
      <selfld name=size>Pick a size:
        <choice>Miniscule
        <choice>Behemoth
      </selfld>
      </dtacol>
    </region>
  </area>
  <botinst>To exit the application, press F3.
</panel>
```

```
                                Selections

 Pick an item:          |   Pick a color:        |   Pick a size:
 __  1.  Widget         |   __  1.  Red          |   __  1.  Miniscule
     2.  Doohickey      |       2.  Green        |       2.  Behemoth
     3.  Gizmo          |                        |

 To exit the application, press F3.
```

Figure 24. Horizontal Region

In the markup for this example, we also changed the format of the DIVIDER tags to provide additional space and a visible line between the selection fields. We did this by specifying TYPE=SOLID and GUTTER=5 on each of the DIVIDER tags. You will notice another characteristic in the preceding example: the divider lines are now vertical. That's because of the manner DTL handles dividers within regions. DTL adheres to the following formatting rules for DIVIDER tags within regions:

- A DIVIDER tag coded within a vertical region formats horizontally.

- A DIVIDER tag coded within a horizontal region formats vertically.

The following markup illustrates how REGION and DIVIDER tags format under different circumstances. This example shows both horizontal and vertical regions, as well as solid and blank dividers.

```
<!doctype dm system>
<panel name=mainpan8>Application
 <topinst>Complete the information below and press Enter.
 <area>
   <dtafld datavar=name entwidth=25 pmtwidth=9>Name
   <dtafld datavar=addr entwidth=25 pmtwidth=9>Address
   <region dir=horiz>
     <dtafld datavar=city pmtwidth=9 entwidth=25>City
     <dtafld datavar=stat pmtwidth=5 entwidth=2>State
     <dtafld datavar=zip pmtwidth=8 entwidth=5>Zip code
   </region>
   <divider type=solid gutter=3>
   <region dir=horiz>
     <selfld name=grade pmtwidth=32 selwidth=33>Highest education level
       <choice>Some high school
       <choice>High school graduate
       <choice>Some college
       <choice>College graduate
       <choice>Some post-graduate work
       <choice>Post graduate degree
     </selfld>
     <divider gutter=5>
     <region>
       <info width=30>
         <p compact>Complete if applicable:
       </info>
       <dtafld datavar=grad pmtwidth=10 entwidth=11>Date of graduation
       <dtafld datavar=field pmtwidth=10 entwidth=11>Field of study
     </region>
   </region>
 </area>
</panel>
```

Figure 25 on page 55 shows how the preceding markup formats.

```
                              Application

Complete the information below and press Enter.

Name  . . _____________________________
Address   _____________________________
City  . . _____________________________  State __  Zip code _____

 ---------------------------------------------------------------------

                                        Complete if applicable:
Highest education level                 Date of
__  1.  Some high school                graduation ___________
    2.  High school graduate            Field of
    3.  Some college                    study  . . ___________
    4.  College graduate
    5.  Some post-graduate work
    6.  Post graduate degree
```

Figure 25. Horizontal Region

This is an example of nesting regions. The data fields for entering the graduation date and field of study are arranged in a vertical region that is nested within a horizontal region.

The ALIGN, DEPTH, EXTEND, INDENT, LOCATION, WIDTH, GRPBOX, and GRPWIDTH attributes allow additional formatting control. The DEPTH and EXTEND attributes are used with scrollable regions. ALIGN, INDENT, LOCATION, and WIDTH affect the placement of fields within the region and the placement of the region within the panel. GRPBOX and GRPWIDTH specify that the region should be displayed as a group box in GUI mode. The optional group box title is supplied as text following the REGION tag ending delimiter. When displayed on a host terminal, a panel defined with a group box will contain the group box title, but will not have a group box border.

Defining a Command Area

Many applications are dependent on a command area in their panels. You define a command area and specify the prompt text of the command area with the CMDAREA tag. The conversion utility supplies the prompt symbol (`===>`) and provides the entry field in the command area for user input.

The conversion utility always formats the command area at the top of the panel. An ISPF run-time option determines the actual display location of the command line.

The command area contains an entry field and command prompt text, and is normally displayed at the bottom of an application panel. Users can enter commands in the command entry field. All commands entered into the command entry field are validated against the commands you define within the application command table and the ISPF-provided commands. For more information on defining the application command table, see "Chapter 8. The Application Command Table" on page 165.

```
<!doctype dm system (<!entity actnbar system>)>
<panel name=cmdxmp1>Application Name
  &actnbar;
  <topinst>Sample command area panel
  <area>
  </area>
  <CMDAREA>
</panel>
```

Here's how the command area displays on the panel:

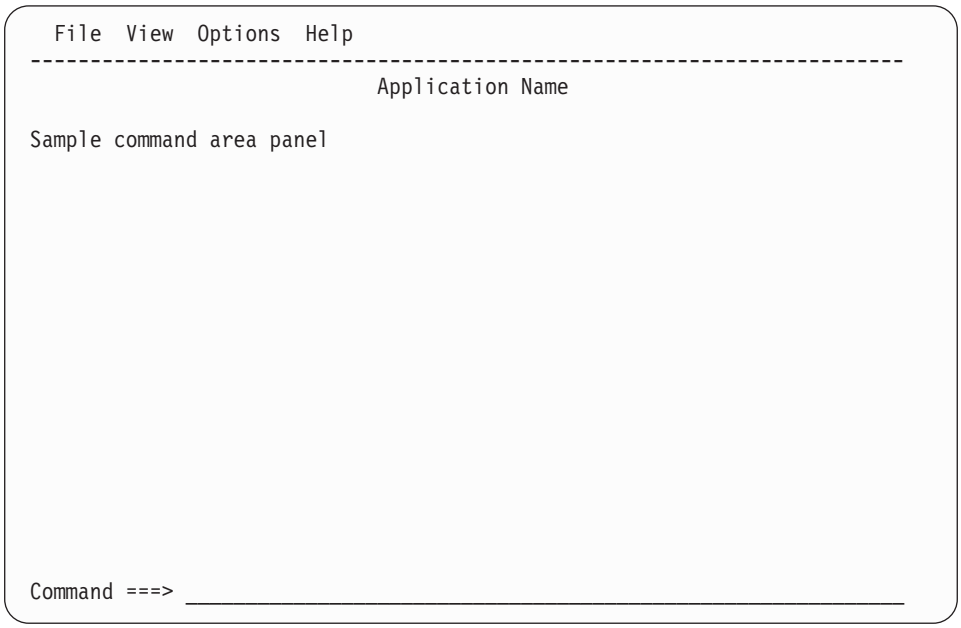


Figure 26. Command Area

In Figure 26 we did not specify the text of the command prompt, so the conversion utility automatically added the text "Command" (or its translated equivalent), which is the default text. If we wanted to specify something other than this text, we could have coded it as tag text, as in this example:

```
<!doctype dm system (<!entity actnbar system>)>
<panel name=cmdxmp2>Application Name
  &actnbar;
  <topinst>Sample command area panel
  <area>
  </area>
  <cmdarea>Enter a command
</panel>
```

You can code up to 59 bytes of prompt text on a standard 76 byte width panel when overriding the default text. Here's how the command prompt looks now:

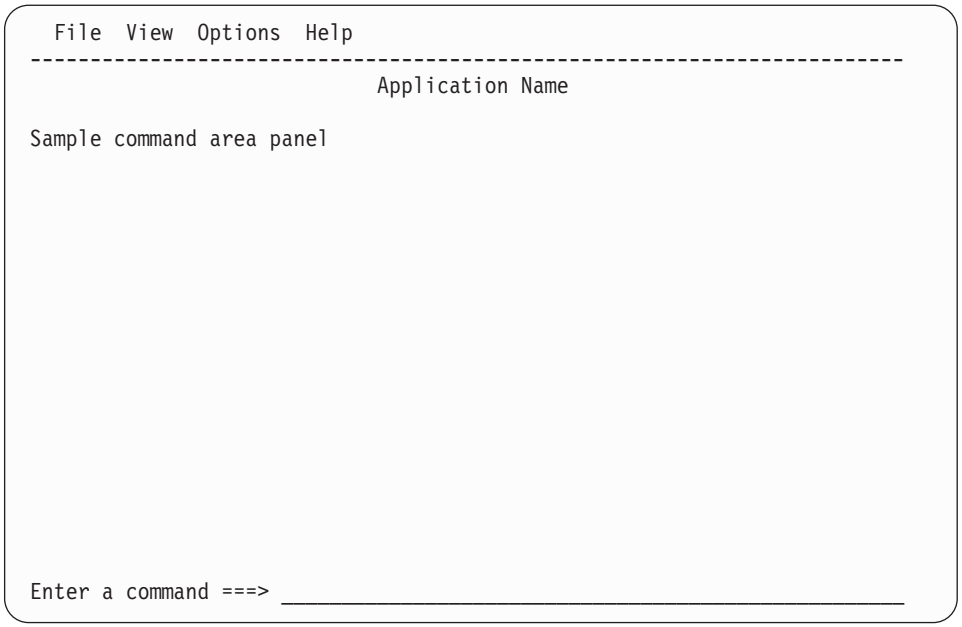


Figure 27. Command Area

Data entered on the command line can be forced to uppercase either by specifying CAPS = ON or by including a VARCLASS tag to define the command area and an XLATL tag to specify translation to uppercase. (The type attribute defines the space available on a standard 76 character width panel using the default command prompt.)

```
<varclass name=vccmd type='char 59'>
  <xlatl format=upper>
  </xlatl>
</varclass>

<varlist>
  <varclass name=zcmd varclass=vccmd>
</varclass>
```

The AUTOTAB, CAPS, CMDLEN, CMDLOC, ENTWIDTH, IMAPNAME, IMAPNAMEP, NAME, NOINIT, NOJUMP, OUTLINE, PAD, PADC, PLACE, PMTTEXT, PSBUTTON, PSVAL, PSVAR, SCRCAPS, SCROLLTAB, SCROLLVAR, and SCRVHELP are attributes that control formatting, initialization, and presentation of the command area.

Defining Panel Defaults

DTL provides you with a tag that makes it easier for you to define attributes and values that are common for multiple application panels: the PANDEF (panel default) tag. This tag must be coded in the source file before any panels it is providing defaults for.

The default PANEL values you can define with the PANDEF tag are:

- The panel dimensions (DEPTH and WIDTH)
- The help panel
- The key mapping list
- The KEYLTYPE value
- The CCSID number

- The WINDOW value
- The WINTITLE value
- The APPTITLE value
- The PAD value
- The PADC value
- The OUTLINE value
- The EXPAND value.
- The MERGESAREA value
- APPLID value
- ENTKEYTEXT value
- IMAPNAME value
- IMAPROW value
- IMAPCOL value
- TMARGIN value
- BMARGIN value

You can use a PANDEF tag to define all of these values, or some of them. You can also override a specific panel default value for a referencing panel by specifying the attribute on the PANEL tag.

For instance, if you create a series of panels that all have the same dimensions and that all refer to the same help panel and key mapping list, you can define these values in a PANDEF definition, and refer to that definition in each of the application panels that use those values. The DTL compiler does the rest of the work for you, as long as the default definition is available as part of the same source file as the panels that refer to it.

For example, if you are creating a series of panels that all share the same values, you could create a PANDEF definition like this:

```
<!doctype dm system>

<pandef id=printdef help=prnthlp depth=20 width=70 keylist=printkey>
```

And refer to the panel default like this on all of the panels in that series:

```
<panel name=panel01 pandef=printdef>A Panel
⋮

</panel>

<panel name=panel02 pandef=printdef>Another Panel
⋮

</panel>
```

When you compile this source file, the PANDEF definition provides those values for the panels that refer to the panel default.

You can also use the PANDEF tag to define common values for individual PANEL attributes. For instance, if the only commonality between application panels is the dimensions, you can use a panel default to define the dimensions and refer only to those values in the application panel definitions:

```
<!doctype dm system>

<pandef id=size depth=20 width=70>
```

```
<panel name=panel01 help=help01
keylist=keylsta pandef=size>A Panel
⋮

</panel>

<panel name=panel02 help=help02
keylist=keylstb pandef=size>Another Panel
⋮

</panel>
```

Anytime you want to change the dimensions of the application panels that refer to a panel default, you only have to make the change in one place: in the PANDEF definition.

To override a PANDEF value, you must specify that value in the PANEL definition. The following example contains a panel default that defines both dimensions and a help panel. While all three PANEL definitions refer to the panel default, the panel with the NAME value *panel03* specifies a different help panel, and thus overrides the PANDEF HELP value.

```
<!doctype dm system>

<pandef id=pandef01 depth=20 width=70 help=help01>

<panel name=panel01 pandef=pandef01>
⋮

</panel>

<panel name=panel02 pandef=pandef01>
⋮

</panel>

<panel name=panel03 pandef=pandef01 help=help02>
⋮

</panel>
```

You can also define multiple panel defaults within a single source file, like this:

```
<!doctype dm system>
<pandef id=pandef01 depth=20 width=70 help=help01>

<pandef id=pandef02 depth=10 width=50 help=help02 keylist=klist01>

<panel name=panel01 pandef=pandef01>
⋮

</panel>

<panel name=panel02 pandef=pandef02>
⋮

</panel>

<panel name=panel03 pandef=pandef01 help=help02>
```

```
⋮
</panel>
```

Chapter 4. Variables and Variable Classes

Much of the information displayed within dialog elements is derived directly from the tags used to define it. Other information is obtained dynamically when the application is running, such as:

- Data that the user supplies
- Data that the application supplies
- Data that ISPF supplies.

In all of these cases, the data is derived from values specified in variables.

DTL provides you with tags to *declare* variables and to define the characteristics of these variables using variable classes. Variables and variable classes are considered global because they can be referred to by more than one element within the same source file. All variables referred to by dialog elements should be declared. Variable names and variable classes should be used consistently throughout dialog elements that are used in the same application.

Variables declared using DTL are accessible to your application through the dialog variable pools and variable services provided by ISPF. Within ISPF display processing, all variable values are in character format. ISPF transforms display variables between their dialog program format and internal display processing character format when retrieving and storing variable values.

Compatibility Considerations

Although the conversion utility processes all of the variable information provided in your DTL source file and issues suppressible warning messages for missing VARDCL tags during the processing of several other tags, such as DTAFLD and LSTCOL, ISPF does not require any of the tags described in the chapter to generate a valid ISPF panel.

The conversion utility supports the SOURCE tag as an alternative means of placing variable processing and validation statements directly into the ISPF panel.

Declaring Variables

You declare variables for dialog elements by coding variable declarations within a variable list. You also specify the variable class associated with each declared variable.

The variable list (VARLIST) tag and its required end tag define the variable list. You code the variable list after any variable classes and before any other tags.

To declare variables, use VARDCL (variable declaration) tags within the VARLIST definition. The VARDCL tag has two required attributes, NAME and VARCLASS.

NAME The NAME attribute specifies the name of the variable used within the DTL source file.

© Copyright IBM Corp. 1989, 2001

For example, a data field definition includes a variable name in the DATAVAR attribute to specify the variable that receives data when the user enters data in the field.

VARCLASS The VARCLASS attribute specifies the variable class associated with the variable declaration. Variable classes define the format and length of variable data plus translations and checks to perform on the data.

In this example, the variable list contains two variable declarations referred to by the data fields in the application panel.

```
<!doctype dm system>

<varclass name=authorc type='char 40'>
<varclass name=catnumc type='char 10'>

<varlist>
  <vardcl name=author varclass=authorc>
  <vardcl name=catnum varclass=catnumc>
</varlist>

<panel name=books1>Book Title Search
  <area>
    <dtacol pmtwidth=20>
      <dtafld entwidth=40 datavar=author>Author
      <dtafld entwidth=10 datavar=catnum>Catalog number
    </dtacol>
  </area>
</panel>
```

Note:

The ISPF Dialog Tag Language conversion utility does not require that you code the VARCLASS, VARDCL, or VARLIST tags for a successful generation of a panel, command table, or message member that includes variables. If the conversion utility finds a variable that does not have an associated VARDCL definition, it issues a suppressible warning message.

The use of the VARCLASS, VARDCL, and VARLIST tags is required if you want to use the facilities provided by the CHECKL and XLATL tags.

Defining Variable Classes

To complete the preceding example, we must code the variable classes that are referred to with the VARDCL VARCLASS attributes. The variable class information must be defined if the conversion utility is to generate)INIT and)PROC section statements for variable translations and validations. (“Variable Validation” on page 65 tells you how to define translations and validity checks.)

The VARCLASS (variable class) tag defines a variable class. You include variable classes in the same source file as the dialog elements and variable list that refer to them. Additionally, you must code variable classes in the source file before the variable list and dialog element definitions. You do this by coding variable classes following the DOCTYPE statement or by coding this information in an external file and embedding the file following the DOCTYPE statement.

There are two required attributes associated with the VARCLASS tag: NAME and TYPE.

NAME
The NAME attribute is used to identify and refer to the variable class.

TYPE The TYPE attribute is used to define the format and entry-field width for variable data.

In addition to the required attributes described above, the VARCLASS tag has an optional MSG attribute. This attribute specifies the message to be displayed if the variable fails any defined validity checks and no message is defined for the XLATL or CHECKL tags. "Chapter 7. Messages" on page 159 tells you how to define messages.

Variable Class Types

DTL supports character variables and numeric variables. In addition, the conversion utility uses the length specified in the TYPE attribute value of the VARCLASS tag to determine the entry width of fields associated with the VARCLASS tag if this width is not defined with the tag used to create the field. For more information on defining field entry widths, see "Chapter 5. Application Panel Fields" on page 83.

Character Variables

You can specify whether single-byte characters, double-byte characters, or mixed double- and single-byte characters are permitted, as well as the maximum number of bytes the variable can accept. Each type is described below:

Type	Description
'CHAR maximum-length'	Specifies a single-byte character string.
'DBCS maximum-length'	Specifies a double-byte character string. The maximum length must be an even number.
'MIXED maximum-length'	Specifies a character string containing single-byte characters, double-byte characters, or both. **Note:** This type is treated as CHAR if the system does not support double-byte characters.
'ANY maximum-length'	Specifies a character string containing single-byte characters, double-byte characters, or both. It is processed by the conversion utility as MIXED.
'EBCDIC maximum-length'	Specifies a single-byte character string.
'%varname maximum-length'	Specifies that a variable name will be used for TYPE in the Panel definition. The application must ensure a valid type is set before the panel is displayed.
'VMASK maximum-length'	A VEDIT statement is added to the generated panel. The 'maximum-length' is the default length for associated variables. The application must use the VMASK service with a user-specified mask value.
'ITIME'	A VEDIT statement is added to the generated panel for associated variables. The default length for the variables is 5. The application must use the VMASK service with a mask of ITIME.

'STDTIME'	A VEDIT statement is added to the generated panel for associated variables. The default length for the variables is 8. The application must use the VMASK service with a mask of STDTIME.
'IDATE'	A VEDIT statement is added to the generated panel for associated variables. The default length for the variables is 8. The application must use the VMASK service with a mask of IDATE.
'STDDATE'	A VEDIT statement is added to the generated panel for associated variables. The default length for the variables is 10. The application must use the VMASK service with a mask of STDDATE.
'JDATE'	A VEDIT statement is added to the generated panel for associated variables. The default length for the variables is 6. The application must use the VMASK service with a mask of JDATE.
'JSTD'	A VEDIT statement is added to the generated panel for associated variables. The default length for the variables is 8. The application must use the VMASK service with a mask of JSTD.

We'll add the variable classes for *authorc* and *catnumc* to the markup example on page 62. We will assume that an author's last name has a maximum of 40 characters and that a catalog number is 10 characters.

```
<!doctype dm system>

<varclass name=authorc type='char 40'>
<varclass name=catnumc type='char 10'>

<varlist>
  <vardcl name=author varclass=authorc>
  <vardcl name=catnum varclass=catnumc>
</varlist>

<panel name=books2>Book Title Search
  <area>
    <dtacol pmtwidth=20>
      <dtafld entwidth=40 datavar=author>Author
      <dtafld entwidth=10 datavar=catnum>Catalog number
    </dtacol>
  </area>
</panel>
```

Numeric Variables

You can use the NUMERIC type to ensure that a valid number is entered in the associated field. You can specify the total number of digits (up to 16) allowed in the number and the number of fractional digits allowed. The conversion utility will generate a VER(variable ENUM) statement for input validation.

For example, the following variable class specifies that the data entered in the associated field has a maximum number of five digits, two of which are fractional.

```
<varclass name=pricevar type='numeric 5 2'>
```

If you do not specify an entry width with the tag that defines the associated field, the conversion utility will calculate an entry width for the field based on the NUMERIC value and allow for a sign, thousands separators, and a decimal point.

Variable Validation

DTL allows you to define translate lists and validity checks as part of the variable class definition by using tags nested within the VARCLASS tag. These built-in translations and checks are especially useful because ISPF automatically performs them on variable values, so the dialog application doesn't need to.

Note: Translations and checks are performed only on variable values that are intended for display. For instance, before displaying the data from a variable specified on the DATAVAR attribute of a DTAFLD tag, ISPF performs any specified translations on the variable retrieved from the application to construct the correct display value. However, ISPF does not perform translations on a variable specified as the CHECKVAR attribute of a CHOICE tag.

Translate Lists

Translate lists provide a means of translating a displayed variable value into a different dialog variable pool value, and vice versa. Translation can occur on input (when the user enters a value), on output (the value stored in the variable pool is translated before the user sees it), or both. This is based on the USAGE value of the tag that refers to a variable using a variable class with translate lists.

To associate a translate list with a variable class, code the XLATL (translate list) tag and its required end tag within the VARCLASS definition.

The type of translation is determined by the value assigned to the FORMAT attribute of the XLATL tag. The two types of translations supported are:
- Uppercase translation
- Item translation

There is an optional MSG attribute on the XLATL tag that allows you to specify your own message to display when input translation specified by the XLATL does not result in a match. For information about defining your own messages, see "Chapter 7. Messages" on page 159.

Upper

Allows you to translate a value to uppercase. To specify this translation, code FORMAT=UPPER on the XLATL tag. This translation is always successful.

We'll add a translate list to the *authorc* variable class in the example on page 64. The translate list converts the author's name to uppercase.

```
<varclass name=authorc type='CHAR 40' msg=liba001>
  <xlatl format=upper>
  </xlatl>
```

The following figure shows the results on input and output translations for the previous example.

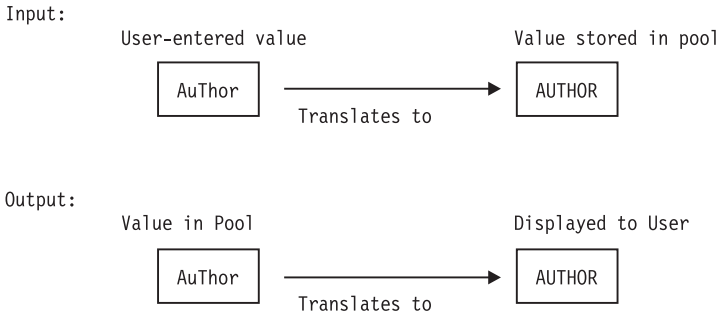


Figure 28. Variable Translation Results

Item Translation

Allows you to translate an internal variable value to a displayed value (or vice-versa) on an item-for-item basis. To specify this translation, either code FORMAT=NONE on the XLATL tag or omit the FORMAT attribute because this is the default. You define the list of possible internal values and the corresponding display values they should be translated to, or from, using the XLATI (translate item) tags nested within the XLATL tag.

To specify an internal value (the value in the variable pool) for a translate item, use the VALUE attribute on the XLATI tag. The XLATI tag text specifies what the user sees (for output) and enters (for input).

The display value is the XLATI tag text. If a display value of all blanks or a display value in which leading, trailing, or embedded blanks are preserved is desired, use the literal (LIT) tag and its required end tag to indicate that blanks are significant.

An explicit match is achieved during translation processing as follows:

- On input, an explicit match occurs when the value the user enters matches one of the specified display values in the translate list. An explicit match also occurs when a display value is omitted (indicating any value is acceptable) and the corresponding internal value is specified.
- On output, an explicit match occurs when the value from the variable pool matches one of the specified internal values in the translate list. An explicit match also occurs when an internal value is omitted (indicating any value is acceptable) and the corresponding display value is specified.

Omitting both the internal value and the display value does not produce an explicit match. This case is discussed later in this section.

Translate list processing is case-sensitive. To ensure that a match results when the user enters the correct display value but in a different or mixed case, code an uppercase conversion translate list before the value translate list.

In the following example, we define the variable class *dayc* to use an internal value for days of the week that is different from the display value. We also ensure that comparisons will be on uppercase values by defining a translate list with FORMAT=UPPER before the item translation list.

```
<!doctype dm system>
```

```
<varclass name=dayc type='CHAR 9'>
  <xlatl format=upper>
  </xlatl>
  <xlatl msg=liba004>
    <xlati value=1>SUNDAY
    <xlati value=2>MONDAY
    <xlati value=3>TUESDAY
    <xlati value=4>WEDNESDAY
    <xlati value=5>THURSDAY
    <xlati value=6>FRIDAY
    <xlati value=7>SATURDAY
  </xlatl>
```

The following figure shows how variable values of variable class *dayc* are translated on input and output.

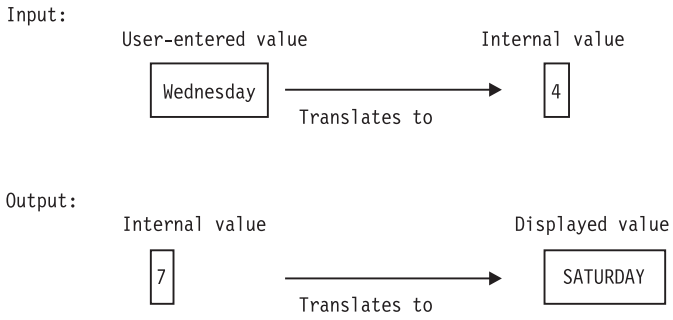


Figure 29. Variable Translation

The previous example shows one translate list with a finite number of translation items. This example assumes that the only possible internal values are 1–7 and the only possible display values are the days of the week. For input fields, a match must be found in this list, or the translation fails and message *liba004* is displayed to the user.

To allow a nonmatching value to be passed on for further processing (either to another translate list or to the validity checks that follow), you code an XLATI tag without an *internal-value* or a *display-value* to indicate that any value is acceptable, as in the following example.

```
<!doctype dm system>

<varclass name=dayc type='CHAR 9'>
  <xlatl format=upper>
  </xlatl>
  <xlatl>
    <xlati value=1>SUNDAY
    <xlati value=2>MONDAY
    <xlati value=3>TUESDAY
    <xlati value=4>WEDNESDAY
    <xlati value=5>THURSDAY
    <xlati value=6>FRIDAY
    <xlati value=7>SATURDAY
    <xlati>
  </xlatl>
```

Because multiple translate lists are permitted, we can expand this example to accept either the days of the week spelled out or their accepted abbreviations.

Because the last XLATI tag in the first translate list has no internal or displayed value, the input value will be passed on for further translate list or validity checking.

```
<!doctype dm system>

<varclass name=dayc type='CHAR 9'>
  <xlatl format=upper>
  </xlatl>
  <xlatl>
    <xlati value=1>SUNDAY
    <xlati value=2>MONDAY
    <xlati value=3>TUESDAY
    <xlati value=4>WEDNESDAY
    <xlati value=5>THURSDAY
    <xlati value=6>FRIDAY
    <xlati value=7>SATURDAY
    <xlati>
  </xlatl>
  <xlatl>
    <xlati value=1>SUN
    <xlati value=2>MON
    <xlati value=3>TUES
    <xlati value=4>WED
    <xlati value=5>THUR
    <xlati value=6>FRI
    <xlati value=7>SAT
  </xlatl>
```

It is possible to omit only the internal value to indicate that any internal value is acceptable. This affects input and output translate processing differently. When translating on input, the value is not translated before being stored in the variable pool. When translating on output, any value not already matched is translated to the displayed value.

In the following example, the *branchc* variable class illustrates translate processing when only the internal value is omitted.

```
<!doctype dm system>

<varclass name=branchc type='CHAR 3'>
  <xlatl format=upper>
  </xlatl>
  <xlatl>
    <xlati value=1>RAL
    <xlati>CRY
  </xlatl>
```

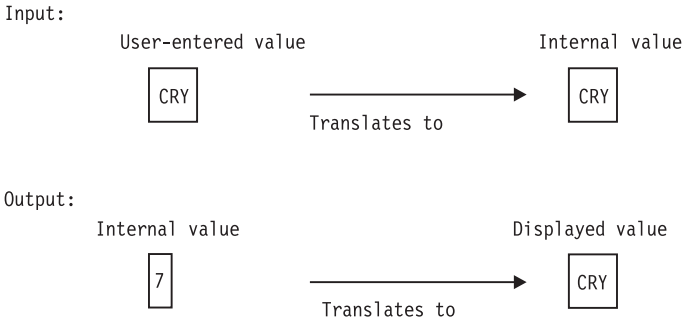


Figure 30. Variable Translation

It is also possible to omit only the display value to indicate that any display value is acceptable. This affects input and output translate processing differently. When translating on input, any value not already matched is translated to the internal value. When translating on output, the internal value is not translated before it is displayed.

We'll change the *branchc* variable class to illustrate translate processing when only the display value is omitted.

```
<!doctype dm system>

<varclass name=branchc type='CHAR 3'>
  <xlatl format=upper>
  </xlatl>
  <xlatl>
    <xlati value=1>RAL
    <xlati value=2>
  </xlatl>
```

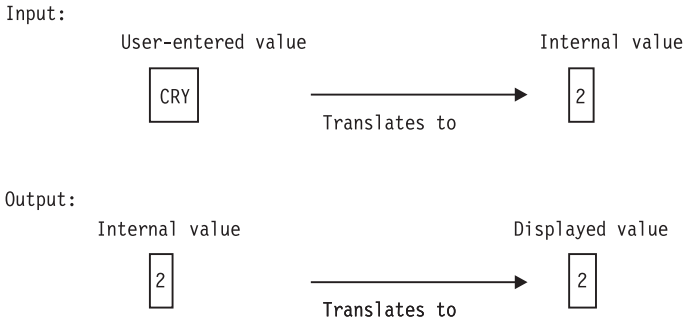


Figure 31. Variable Translation

It is possible to specify that less than the full input value be entered by the use of the TRUNC attribute. Output translation is not affected.

We'll change the *branchc* variable class again to illustrate.

```
<!doctype dm system>

<varclass name=branchc type='CHAR 3'>
  <xlatl format=upper>
  </xlatl>
  <xlatl format=none trunc=1>
    <xlati value=1>RAL
    <XLATI VALUE=2>
  </xlatl>
```

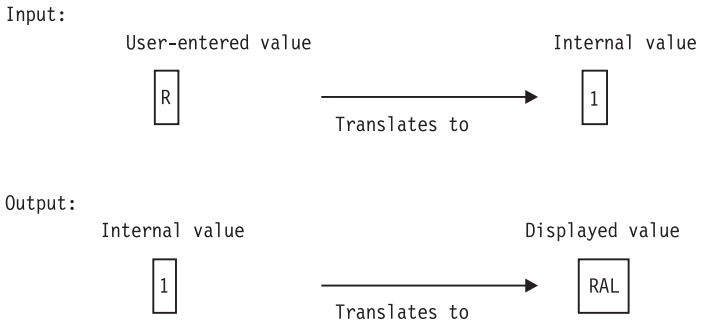


Figure 32. Variable Translation

Validity Checks

You use validity checks to automatically verify data input by the user. Validity checks are coded after any translate lists.

To associate a validity check with a variable class, code the CHECKL (check list) tag and its required end tag either following the last translate list, or if no translate list exists, following the VARCLASS start tag. The individual check item that defines the test to perform is coded using the CHECKI (check item) tag nested within the check list. You can code one CHECKI tag in a CHECKL definition. However, you can code multiple CHECKL tags within a variable class definition.

There is an optional MSG attribute on the CHECKL tag that allows you to specify your own message to display when the entered value fails the test. If you do not specify a message, ISPF Dialog Manager supplies a default message for you. For more information about defining your own messages, see "Chapter 7. Messages" on page 159.

A value entered by the user must pass the check item test for the check list to be considered successful. Furthermore, because there can be multiple check lists defined, all check lists must be successful for the validation to be successful.

The TYPE attribute of the CHECKI tag allows you to specify the various validity tests of the input. You can define the following types of validity checks:

- RANGE
- ALPHA
- VALUES
- VALUESX
- CHARS (limited to characters for HEX, BIT, and NUM tests)
- FILEID
- DSNAME
- DSNAMEF
- DSNAMEFM
- DSNAMEPQ
- DSNAMEQ
- NAME
- NAMEF
- DBCS
- EBCDIC
- MIX
- ALPHAB

- PICT
- PICTCN
- LISTV
- LISTVX
- LEN
- ENUM
- BIT
- NUM
- HEX
- INCLUDE
- IDATE
- STDDATE
- JDATE
- JSTD
- ITIME
- STDTIME
- IPADDR4

RANGE

To perform a range test, specify the check item TYPE attribute as RANGE. A range check allows you to check a value within a numeric range including the end points. The PARM1 attribute specifies the lower bound; PARM2 specifies the upper bound. The range delimiters can include 16 digits, and may contain a preceding sign (– or +).

The following example of a NUMERIC variable class contains a range check that ensures that catalog numbers are in the range 50 to 90000000.

```
<!doctype dm system>

<varclass name=catnumc type='NUMERIC 8'>
  <checkl msg=liba005>
    <checki type=range parm1=50 parm2=90000000>
  </checkl>
```

The conversion utility generates an ISPF range verification statement in the)PROC section.

ALPHA

To perform an alphabetic test, specify the check item TYPE attribute as ALPHA. An alpha check limits the characters allowed to A–Z, a–z, #, $, and @.

The following example of an alpha check in the *authorc* variable class ensures that authors' names are alphabetic.

```
<!doctype dm system>

<varclass name=authorc type='CHAR 40'>
  <checkl msg=liba006>
     <checki type=alpha>
  </checkl>
```

The conversion utility generates an ISPF alpha verification statement in the)PROC section.

VALUES

To perform a values test, specify the check item TYPE attribute as VALUES. A values test allows you to specify a list of values. The value the user enters must

match one of the values in the list. The PARM1 attribute must have the value EQ. The PARM2 attribute specifies the list of values. Because case is respected in a VALUES check, if you want case to be ignored, you must code an UPPER translation and code the values all in uppercase.

The following example of a check in a variable class named *subject* ensures that the value entered is MATH, SCIENCE, ENGLISH, or HISTORY.

```
<!doctype dm system>

<varclass name=subject type='char 10'>
  <xlatl format=upper>
  </xlatl>
  <checkl msg=liba008>
     <checki type=values parm1=eq
     parm2='MATH SCIENCE ENGLISH HISTORY'>
  </checkl>
```

The conversion utility generates a LIST verification statement in the)PROC section.

VALUESX

The check item type VALUESX is the opposite of VALUES. This test allows you to specify a list of values that are not valid. The PARM1 attribute must have the value NE. The PARM2 attribute specifies the list of values that are not valid. The value the user enters cannot match any of the values specified in the list. Because case is respected in a VALUESX check, if you want case to be ignored, you must code an UPPER translation and code the values all in uppercase.

The following example of a check in a variable class named *subject* ensures that the value entered is not MATH, SCIENCE, ENGLISH, or HISTORY.

```
  <!doctype dm system>

  <varclass name=subject type='char 10'>
    <xlatl format=upper>
    </xlatl>
    <checkl msg=liba008>
       <checki type=valuesx parm1=ne
       parm2='MATH SCIENCE ENGLISH HISTORY'>
    </checkl>
```

The conversion utility generates a LISTX verification statement in the)PROC section.

CHARS

The conversion utility supports BIT, HEX and NUM validation with TYPE=CHARS. The PARM1 attribute must have the value EQ. The PARM2 attribute value can be either “01” for BIT validation, “0123456789ABCDEFabcdef” for HEX validation, or “0123456789” for NUM validation.

The following example of a check list in the *hexc* variable class validates hexadecimal values.

```
<!doctype dm system>

<varclass name=hexc type='char 2'>
  <checkl msg=liba008>
     <checki type=chars parm1=eq parm2='0123456789ABCDEFabcdef'>
  </checkl>
```

The conversion utility generates an ISPF hex verification statement in the)PROC section.

FILEID

To perform a FILEID test, specify the check item TYPE attribute as FILEID.

The following example of a FILEID check in the *infile* variable class ensures that specified variables contain a valid file ID in CMS syntax.

```
<!doctype dm system>

<varclass name=infile type='CHAR 20'>
  <checkl msg=liba010>
     <checki type=fileid>
  </checkl>
```

The conversion utility generates an ISPF FILEID verification statement in the)PROC section.

DSNAME

To perform a DSNAME test, specify the check item TYPE attribute as DSNAME.

The following example of a DSNAME check in the *namefile* variable class ensures that the specified variables contain a valid TSO file name.

```
<!doctype dm system>

<varclass name=namefile type='CHAR 44'>
  <checkl msg=liba011>
     <checki type=dsname>
  </checkl>
```

The conversion utility generates a DSNAME verification statement in the)PROC section.

DSNAMEF

To perform a DSNAMEF test, specify the check item TYPE attribute as DSNAMEF.

The following example of a DSNAMEF check in the *namefile* variable class ensures that the specified variables contain a valid TSO file name.

```
<!doctype dm system>

<varclass name=namefile type='CHAR 44'>
  <checkl msg=liba011>
     <checki type=dsnamef>
  </checkl>
```

The conversion utility generates a DSNAMEF verification statement in the)PROC section.

DSNAMEFM

To perform a DSNAMEFM test, specify the check item TYPE attribute as DSNAMEFM.

The following example of a DSNAMEFM check in the *namefile* variable class ensures that the specified variables contain a valid TSO file name.

```
<!doctype dm system>
```

```
<varclass name=namefile type='CHAR 44'>
  <checkl msg=liba011>
     <checki type=dsnamefm>
  </checkl>
```

The conversion utility generates a DSNAMEFM verification statement in the)PROC section.

DSNAMEPQ

To perform a DSNAMEPQ test, specify the check item TYPE attribute as DSNAMEPQ.

The following example of a DSNAMEPQ check in the *namefile* variable class ensures that the specified variables contain a valid TSO file name.

```
<!doctype dm system>

<varclass name=namefile type='CHAR 44'>
  <checkl msg=liba011>
     <checki type=dsnamepq>
  </checkl>
```

The conversion utility generates a DSNAMEPQ verification statement in the)PROC section.

DSNAMEQ

To perform a DSNAMEQ test, specify the check item TYPE attribute as DSNAMEQ.

The following example of a DSNAMEQ check in the *namefile* variable class ensures that the specified variables contain a valid TSO file name.

```
<!doctype dm system>

<varclass name=namefile type='CHAR 44'>
  <checkl msg=liba011>
     <checki type=dsnameq>
  </checkl>
```

The conversion utility generates a DSNAMEQ verification statement in the)PROC section.

NAME

To perform a NAME test, specify the check item TYPE attribute as NAME.

The following example of a NAME check in the *chapters* variable class ensures that the variable contains a valid name, following the rules of member names.

```
<!doctype dm system>

<varclass name=chapters type='CHAR 8'>
  <checkl msg=liba012>
     <checki type=name>
  </checkl>
```

The conversion utility generates a NAME verification statement in the)PROC section.

NAMEF

To perform a NAMEF test, specify the check item TYPE attribute as NAMEF.

The following example of a NAMEF check in the *chapters* variable class ensures that the variable contains a valid name, following the rules of member names.

```
<!doctype dm system>

<varclass name=chapters type='CHAR 8'>
  <checkl msg=liba012>
     <checki type=namef>
  </checkl>
```

The conversion utility generates a NAMEF verification statement in the)PROC section.

DBCS

To perform a DBCS test, specify the check item TYPE attribute as DBCS.

The following example of a DBCS check in the *dbdesc* variable class ensures that specified variables contain valid DBCS characters.

```
<!doctype dm system>

<varclass name=dbdesc type='DBCS 12'>
  <checkl msg=liba013>
     <checki type=dbcs>
  </checkl>
```

The conversion utility generates a DBCS verification statement in the)PROC section.

EBCDIC

To perform an EBCDIC test, specify the check item TYPE attribute as EBCDIC.

The following example of an EBCDIC check in the *title* variable class ensures that specified variables contain valid EBCDIC characters.

```
<!doctype dm system>

<varclass name=title1 type='CHAR 40'>
  <checkl msg=liba014>
     <checki type=ebcdic>
  </checkl>
```

The conversion utility generates an EBCDIC verification statement in the)PROC section.

MIX

To perform a MIX test, specify the check item TYPE attribute as MIX.

The following example of a MIX check in the *title2* variable class ensures that specified variables contain valid DBCS and EBCDIC characters.

```
<!doctype dm system>

<varclass name=title2 type='CHAR 40'>
  <checkl msg=liba015>
     <checki type=mix>
  </checkl>
```

The conversion utility generates a MIX verification statement in the)PROC section.

ALPHAB

To perform an ALPHAB test, specify the check item TYPE attribute as ALPHAB. An ALPHAB check limits the characters allowed to A–Z or a–z. Blanks are not allowed.

The following example of an ALPHAB check in the *chapters* variable class ensures that chapters' names are alphabetic.

```
<!doctype dm system>

<varclass name=chapters type='CHAR 8'>
  <checkl msg=liba016>
     <checki type=alphab>
  </checkl>
```

The conversion utility generates an ALPHAB verification statement in the)PROC section.

PICT

To perform a PICT check, specify the check item TYPE attribute as PICT. A PICT check allows you to specify a pattern used to validate the variable. The PARM1 attribute must have the value EQ. The PARM2 attribute contains the validation character string.

The following example of a PICT check in the *socsec* variable class validates the format of a social security number.

```
<!doctype dm system>

<varclass name=socsec type='CHAR 11'>
  <checkl msg=liba017>
     <checki type=pict parm1=eq parm2='nnn-nn-nnnn'>
  </checkl>
```

The conversion utility generates a PICT verification statement in the)PROC section.

PICTCN

To perform a PICTCN check, specify the check item TYPE attribute as PICTCN. A PICTCN check allows you to specify a pattern containing required characters to validate the variable. The PARM1 attribute contains a mask character. The PARM2 attribute contains the field-mask. The PARM3 attribute contains the validation string.

The following example of a PICTCN check in the *socsec* variable class validates the format of a social security number, including the hyphen (-) character in positions 4 and 7.

```
<!doctype dm system>

<varclass name=socsec type='CHAR 11'>
  <checkl msg=liba017>
     <checki type=pictcn parm1='*' parm2='***-**-****'
                 parm3='nnn-nn-nnnn'>
  </checkl>
```

The conversion utility generates a PICTCN verification statement in the)PROC section.

LISTV

To perform a LISTV check, specify the check item TYPE attribute as LISTV. A LISTV test allows you to provide a variable name that has been defined by your application to contain a list of valid variable values. The PARM1 attribute must have the value EQ. The PARM2 attribute must be a variable name entered with "%" as the first character.

The following example of a LISTV check in the *majors* variable class validates major subjects, providing the application has previously defined the *listitem* variable to contain the value "MATH SCIENCE ENGLISH HISTORY".

```
<!doctype dm system>

<varclass name=majors type='CHAR 8'>
  <checkl msg=liba018>
     <checki type=listv parm1=eq parm2=%listitem>
  </checkl>
```

The conversion utility generates a LISTV verification statement in the)PROC section.

LISTVX

The check item type LISTVX is the opposite of LISTV. A LISTVX test allows you to provide a variable name that has been defined by your application to contain a list of variable values that are not valid. PARM1 attribute must have the value NE. The PARM2 attribute must be a variable name entered with "%" as the first character.

The following example of a LISTVX check in the *majors* variable class validates major subjects, providing the application has previously defined the *listitem* variable to contain the value "MATH SCIENCE ENGLISH HISTORY".

```
<!doctype dm system>

<varclass name=majors type='CHAR 8'>
  <checkl msg=liba018>
     <checki type=listvx parm1=ne parm2=%listitem>
  </checkl>
```

The conversion utility generates a LISTVX verification statement in the)PROC section.

LEN

To perform a LEN check, specify the check item TYPE attribute as LEN. A LEN test allows you to validate the length of the variable. The PARM1 attribute can be a relational operator or a variable name that contains a relational operator. Valid relational operators are EQ, LT, GT, LE, GE, NE, NG, or NL. If a variable name is used, it must be preceded by a "%". The PARM2 value can be either a number or a variable name that contains the number. If you enter a number, it must be in the range of 1–99999. If you use a variable name, it must be preceded by a "%".

The following example of a LEN check in the *chapters* variable class validates the length of chapter names.

```
<!doctype dm system>

<varclass name=chapters type='CHAR 8'>
  <checkl msg=liba019>
     <checki type=len parm1=le parm2=8>
  </checkl>
```

The conversion utility generates a LEN verification statement in the)PROC section.

ENUM

To perform an ENUM check, specify the check item TYPE attribute as ENUM. An ENUM check allows you to verify a variable as extended numeric. ISPF verifies variable values for correct decimal and comma notation plus correct sign placement.

The following example of an ENUM check in the *quantity* variable class ensures that specified variables are in correct extended numeric format.

```
<!doctype dm system>

<varclass name=quantity type='CHAR 10'>
  <checkl msg=liba020>
     <checki type=enum>
  </checkl>
```

The conversion utility generates an ENUM verification statement in the)PROC section.

BIT

To perform a BIT check, specify the check item TYPE as BIT. A BIT check allows you to verify that a variable contains only 0's and 1's.

The following example of a BIT check in the *choices* variable class ensures that specified variables are in BIT format.

```
<!doctype dm system>

<varclass name=choices type='CHAR 1'>
  <checkl msg=liba021>
     <checki type=bit>
  </checkl>
```

NUM

To perform a NUM check, specify the check item TYPE attribute as NUM. A NUM check allows you to verify a variable as a numeric character 0–9.

The following example of a NUM check in the *numbers* variable class ensures that specified variables are numeric.

```
<!doctype dm system>

<varclass name=numbers type='CHAR 5'>
  <checkl msg=liba022>
     <checki type=num>
  </checkl>
```

HEX

To perform a HEX check, specify the check item TYPE attribute as HEX. A HEX check allows you to specify a variable that contains only hexadecimal characters (0–9, A–F).

The following example of a HEX check in the *hexc* variable class validates hexadecimal values.

```
<!doctype dm system>
```

```
<varclass name=hexc type='CHAR 2'>
  <checkl msg=liba008>
     <checki type=hex>
  </checkl>
```

INCLUDE

To perform an INCLUDE check, specify the TYPE attribute as INCLUDE, and, at a minimum, the PARM2 attribute as ALPHA, ALPHAB, or NUM. The PARM1 and PARM3 attributes are optional.

The following example of an INCLUDE check in the *incl* variable class will allow an imbedded blank and validate the values for both the ALPHA and NUM characters as defined earlier.

```
<!doctype dm system>

<varclass name=incl type='CHAR 10'>
  <checkl msg=liba023>
     <checki type=include parm1=IMBLK parm2=ALPHA parm3=NUM>
  </checkl>
```

Note: Refer to the *ISPF Dialog Developer's Guide and Reference* for additional information concerning panel variable verification.

IDATE

To perform an IDATE check, specify the TYPE attribute as IDATE. An IDATE check allows you to validate an 8 character international date, including the national language date delimiter. The format for the United States is YY/MM/DD.

The following example validates an IDATE.

```
<!doctype dm system>

  <varclass name=idate type='CHAR 8'>
    <checkl msg=liba024>
     <checki type=idate>
    </checkl>
```

STDDATE

To perform an STDDATE check, specify the TYPE attribute as STDDATE. An STDDATE check allows you to validate a 10 character standard date, including the national language date delimiter. The format for the United States is YYYY/MM/DD.

The following example validates an STDDATE.

```
<!doctype dm system>

  <varclass name=stddate type='CHAR 10'>
    <checkl msg=liba025>
     <checki type=stddate>
    </checkl>
```

JDATE

To perform a JDATE check, specify the TYPE attribute as JDATE. A JDATE check allows you to validate a 6 character Julian date. The format is YY.DDD.

The following example validates a JDATE.

```
<!doctype dm system>

  <varclass name=jdate type='CHAR 6'>
    <checkl msg=liba026>
     <checki type=jdate>
    </checkl>
```

JSTD

To perform a JSTD check, specify the TYPE attribute as JSTD. A JSTD check allows you to validate an 8 character Julian date. The format is YYYY.DDD.

The following example validates a JSTD.

```
<!doctype dm system>

  <varclass name=jstd type='CHAR 8'>
    <checkl msg=liba026>
     <checki type=jstd>
    </checkl>
```

ITIME

To perform an ITIME check, specify the TYPE attribute as ITIME. An ITIME check allows you to validate a 5 character international time, including the national language time delimiter. The format for the United States is HH:MM.

The following example validates an ITIME.

```
<!doctype dm system>

  <varclass name=itime type='CHAR 5'>
    <checkl msg=liba027>
     <checki type=itime>
    </checkl>
```

STDTIME

To perform a STDTIME check, specify the TYPE attribute as STDTIME. A STDTIME check allows you to validate an 8 character standard time, including the national language time delimiter. The format for the United States is HH:MM:SS.

The following example validates a STDTIME.

```
<!doctype dm system>

  <varclass name=stdtime type='CHAR 8'>
    <checkl msg=liba028>
     <checki type=stdtime>
    </checkl>
```

IPADDR4

To perform a IPADDR4 check, specify the TYPE attribute as IPADDR4. A IPADDR4 check allows you to verify a 15 character IP address of the format xxx.xxx.xxx.xxx.

The following example validates an IPADDR4.

```
<!doctype dm system>

  <varclass name=ipaddr4 type='CHAR 15'>
    <checkl msg=liba029>
     <checki type=ipaddr4>
    </checkl>
```

Overriding Variable Classes

Some tags, such as DTAFLD, allow you to specify a different variable class for a variable other than the default one that was specified when the variable was declared using the VARDCL tag. This is called an overriding variable class and is used to perform different translates and validity checks from those provided by the default variable class.

Chapter 5. Application Panel Fields

Most of the direct interaction that takes place between the user and the application is through the use of interactive fields. They provide a means for the user to communicate data to the application, as well as receive data from the application.

The type of interaction the user has with the application depends on the task. The task, in turn, determines the fields' characteristics. The appearance of the fields, the application's response to user input, and assistance such as messages and help information must all be considered when defining an interactive field.

In this chapter, we tell you how to use the Dialog Tag Language to define the following types of fields and their operating characteristics:
- Data fields
- Selection fields
- List fields.

This chapter begins with a description of field prompts for data fields and selection fields.

Field Prompts

A field prompt is static, descriptive text that explains the field it is associated with. Data fields and selection fields support the use of field prompts. To define a field prompt for a data field or selection field, specify the prompt text as the tag text on the DTAFLD and SELFLD tags.

The PMTLOC attribute defines the location of the prompt using one of the following values:

PMTLOC = ABOVE
The prompt appears above and left-aligned with the field. This is the default for selection fields.

PMTLOC = BEFORE
The prompt appears directly in front of and on the same line as the field. This is the default for data fields.

You should define the amount of space the prompt uses by specifying the PMTWIDTH attribute on the DTAFLD and SELFLD tags. If the prompt text is longer than the width you specify on PMTWIDTH, the prompt is word-wrapped on multiple lines. Using the PMTWIDTH attribute can ensure that multiple fields with prompts are aligned evenly. If you do not specify PMTWIDTH, the field prompt length is determined by the length of the prompt text.

When PMTLOC=BEFORE, the conversion utility inserts leader dots at the end of the prompt text to fill the specified prompt width. For output-only data fields, a colon is used in place of the last leader dot. For fields with this prompt location, it is a good idea to specify a PMTWIDTH with a value that allows for leader dots after the prompt text. This lends consistency to the panel appearance. The conversion utility issues a warning message when there is insufficient space for leader dots.

© Copyright IBM Corp. 1989, 2001

Figure 33 shows how prompts appear.

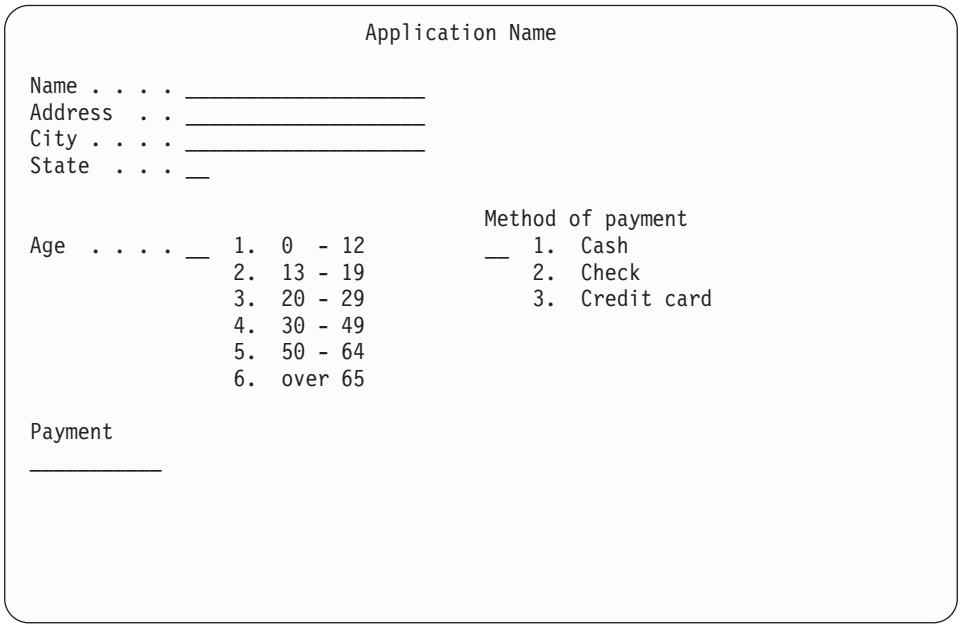


Figure 33. Prompt Locations

The **Name**, **Address**, **City**, and **State** data fields show the prompts in front of the fields (PMTLOC=BEFORE), as does the **Age** field, which shows a prompt for a selection field. The same prompt width is used on the first five fields so that they align evenly. The **Method of payment** and **Payment** fields demonstrate the prompt above the field (PMTLOC=ABOVE).

Here's the markup used to demonstrate the field prompts in Figure 33:

```
<!doctype dm system>

<varclass name=sampcls type ='char 20'>
<varclass name=statcls type='char 2'>
<varclass name=numcls  type='numeric 8 2'>
<varclass name=char1   type='char 1'>

<varlist>
  <vardcl name=name varclass=sampcls>
  <vardcl name=addr varclass=sampcls>
  <vardcl name=city varclass=sampcls>
  <vardcl name=stat varclass=statcls>
  <vardcl name=pay varclass=numcls>
  <vardcl name=age varclass=char1>
  <vardcl name=paymeth varclass=char1>
</varlist>

<panel name=pmt01>Application Name
  <area>
    <dtafld datavar=name entwidth=20 pmtwidth=12>Name
    <dtafld datavar=addr entwidth=20 pmtwidth=12>Address
    <dtafld datavar=city entwidth=20 pmtwidth=12>City
    <dtafld datavar=stat entwidth=2 pmtwidth=12>State
    <divider>
      <region dir=horiz>
        <selfld name=age selwidth=20 pmtloc=before pmtwidth=12>Age
          <choice>0  - 12
```

```
          <choice>13 - 19
          <choice>20 - 29
          <choice>30 - 49
          <choice>50 - 64
          <choice>over 65
        </selfld>
        <divider gutter=5>
        <selfld name=paymeth selwidth=24 pmtwidth=20>Method of payment
          <choice>Cash
          <choice>Check
          <choice>Credit card
        </selfld>
      </region>
    <divider>
    <dtafld datavar=pay entwidth=11 pmtloc=above pmtwidth=7>Payment
  </area>
</panel>
```

Figure 34 shows how the prompt width can affect the appearance of the prompt text.
The prompts in the two data fields are formatted differently. The prompt text of

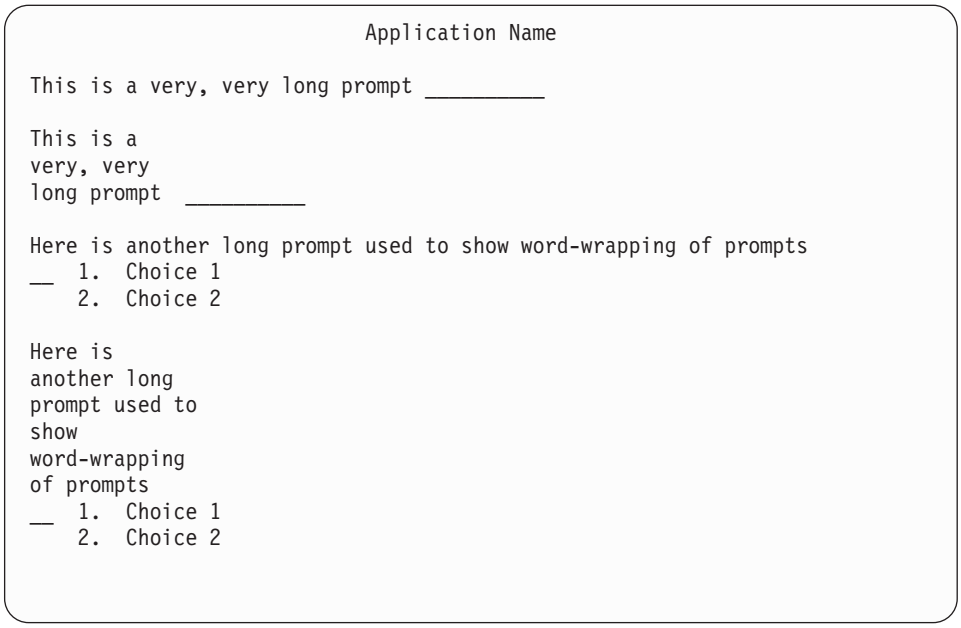


Figure 34. Prompt Widths

the first data field is not wrapped. It formats on one line, using as much space as necessary (up to the maximum available formatting width). The second data field has the same prompt text, with a prompt width that is less than the amount of space needed, so the prompt text is wrapped to as many lines as are needed. Similarly, the two selection fields also demonstrate how the prompt text appears based on the prompt width. The prompt text of data fields and selection fields can be displayed differently by omitting or specifying different values for the PMTWIDTH attribute.

The following is the markup used to demonstrate the field prompts in Figure 34:

```
<!doctype dm system>

<varclass name=sampcls type='char 10'>
<varclass name=char1 type='char 1'>
```

```
<varlist>
  <vardcl name=samplea varclass=sampcls>
  <vardcl name=sampleb varclass=sampcls>
  <vardcl name=samplec varclass=char1>
  <vardcl name=sampled varclass=char1>
</varlist>

<panel name=pmt02>Application Name
  <area>
    <dtacol entwidth=10 selwidth=76>
    <dtafld datavar=samplea>This is a very, very long prompt
    <divider>
    <dtafld datavar=sampleb pmtwidth=12>This is a very, very long prompt
    <divider>
    <selfld name=samplec>Here is another long prompt used to show
            word-wrapping of prompts
      <choice>Choice 1
      <choice>Choice 2
    </selfld>
    <divider>
    <selfld name=sampled pmtwidth=14>Here is another long prompt used to show
                                     word-wrapping of prompts
      <choice>Choice 1
      <choice>Choice 2
    </selfld>
    </dtacol>
  </area>
</panel>
```

Defining Data Fields

Data fields are used to display variable data and to allow the user to enter data. To define a data field, use the DTAFLD tag. Every data field must have an associated variable, which is specified on the required DATAVAR attribute. Like all variables used on the panel, the variable named on the DATAVAR attribute can be declared using the VARDCL tag.

The purpose of the data field is defined using one of these values on the USAGE attribute of the DTAFLD tag:

IN Defines an entry (input-only) data field. An entry data field allows the user to enter data. When an entry field is initially displayed, it is padded with underscore characters, unless the data is right-justified.

OUT Defines an output-only data field. An output-only data field is used to display the current value of the variable associated with the data field. The user cannot tab to or interact with an output-only field.

BOTH Defines an input/output data field. When an input/output field is initially displayed, the current value of the associated variable is displayed, and the user can enter data into the field as well. If you do not specify the USAGE attribute, BOTH is the default.

Data fields support field prompts, which can be placed in front of or above the data field.

This panel contains examples of all three types of data fields:

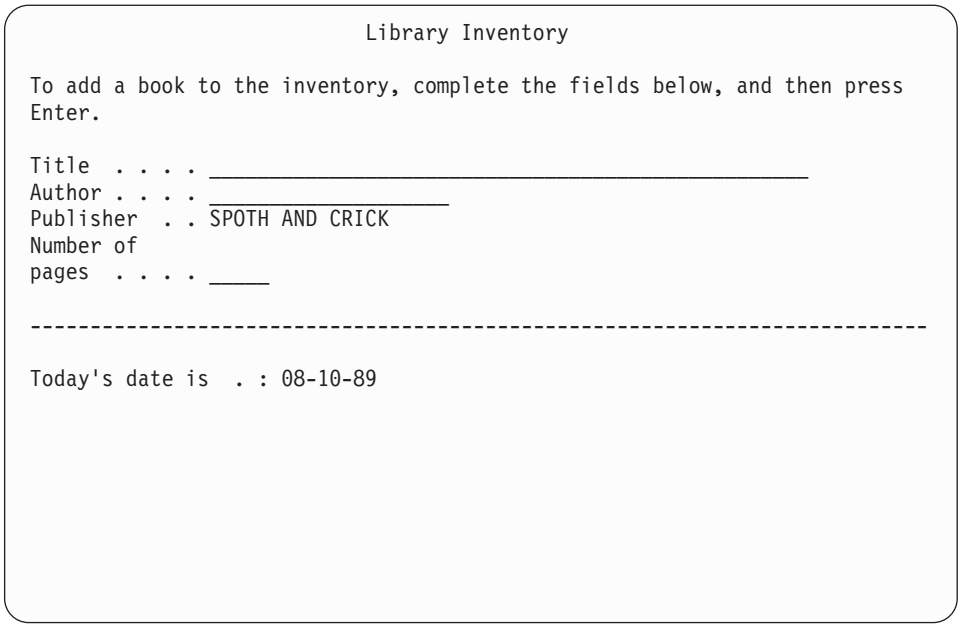


Figure 35. Data Fields

Here's the markup for Figure 35:

```
<!doctype dm system>

<varclass name=titlcls type='char 50'>
<varclass name=bookcls type='char 20'>
<varclass name=pagecls type='numeric 5'>
<varclass name=datecls type='char 8'>

<varlist>
  <vardcl name=title   varclass=titlcls>
  <vardcl name=author  varclass=bookcls>
  <vardcl name=publish varclass=bookcls>
  <vardcl name=pages   varclass=pagecls>
  <vardcl name=curdate  varclass=datecls>
</varlist>

<panel name=dfdxmp1a>Library Inventory
  <topinst>To add a book to the inventory, complete the fields below,
           and then press Enter.
  <area>
    <dtafld datavar=title usage=in pmtwidth=14>Title
    <dtafld datavar=author usage=in entwidth=20 pmtwidth=14>Author
    <dtafld datavar=publish entwidth=20 pmtwidth=14>Publisher
    <dtafld datavar=pages usage=in entwidth=5 pmtwidth=14>Number of pages
    <divider type=solid gutter=3 gap=no>
    <dtafld datavar=curdate usage=out entwidth=8 pmtwidth=20>Today's date is
  </area>
</panel>
```

In the previous example, there are three input-only data fields, an input/output data field, and an output-only data field. The value of the associated variable is not displayed in an input-only data field, so when the panel is initially displayed, the **Title**, **Author**, and **Number of pages** fields are blank. The **Publisher** data field assumes the default, BOTH, so the current value of the associated variable, *publish*, is displayed in the data field when the panel is initially displayed. The output-only data field is used to display the current date. The user cannot interact with this

data field, since it is used only to display variable data. The user can enter data into any of the data fields except the output-only field.

Data Field Width

The width of a data field is determined by the value you specify for the ENTWIDTH attribute of the DTAFLD tag. You should specify ENTWIDTH for all data fields. In the previous example, ENTWIDTH is specified for each DTAFLD tag except for the **Title** field, whose length is determined as discussed next.

If you do not specify a value for ENTWIDTH, the width of the data field is determined by the value specified for the TYPE attribute of the VARCLASS tag associated with the variable named in the DTAFLD DATAVAR attribute. For example, the **Title** field in Figure 35 on page 87 has an entry width of 50 as determined by the variable class *titlcls*, which has the TYPE value "char 50". This variable class is associated with the data field through the variable declaration *title*, which is specified as the data field's DATAVAR attribute value. For more information on variables and variable classes, see "Chapter 4. Variables and Variable Classes" on page 61.

The formatted width of the field is 2 positions more than the ENTWIDTH value to provide for an attribute byte both before and after the field. The maximum width for an entry field is the remaining available formatting width in the panel.

Note: The conversion utility tracks the remaining width available for use. For data fields, the width of the entry field has first priority, followed by the prompt width, and then the description width.

Data Field Descriptions

In addition to a field prompt, you can provide additional descriptive text for a data field using the DTAFLDD (data field description) tag. You code the DTAFLDD tag following the definition of the data field being described. The DTAFLDD tag has no attributes or required end tag. Multiple data field descriptions can be coded if necessary, and each description begins a new line.

The data field description appears to the right of the entry field, taking up as much room as is available, unless you have used the DESWIDTH attribute of the DTAFLD tag to specify a width for the description. If the DESWIDTH attribute is defined, the data field description is displayed within the description width specified (or defaulted), and word-wrapped on multiple lines, if necessary.

This panel contains data field descriptions.

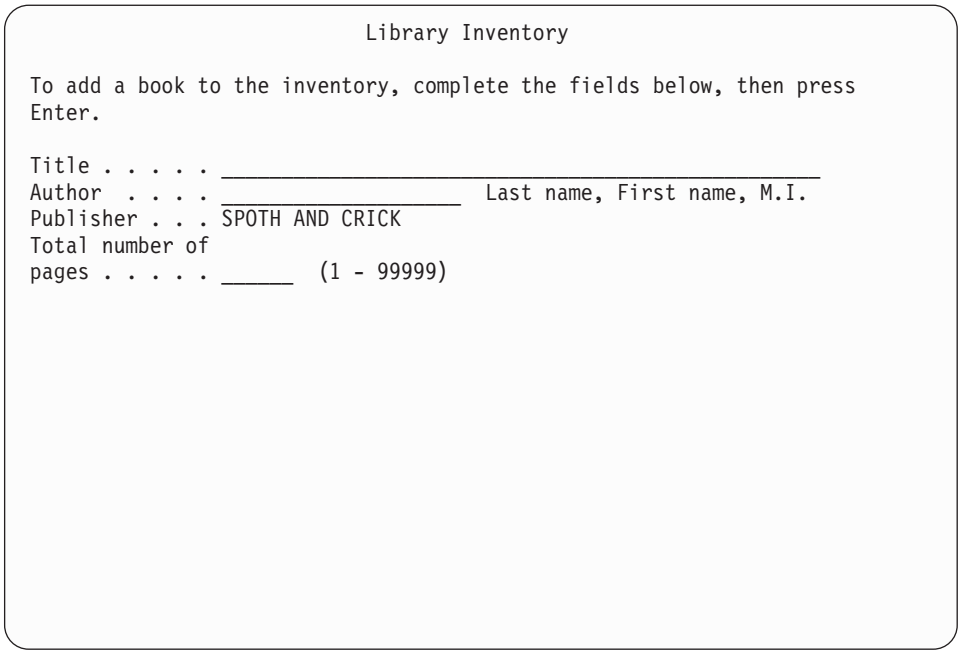


Figure 36. Data Field Description

Here's the markup used to generate the panel in Figure 36:

```
<!doctype dm system>

<varclass name=titlcls type='char 50'>
<varclass name=bookcls type='char 20'>
<varclass name=pagecls type='numeric 5'>

<varlist>
  <vardcl name=title   varclass=titlcls>
  <vardcl name=author  varclass=bookcls>
  <vardcl name=publish varclass=bookcls>
  <vardcl name=pages   varclass=pagecls>
</varlist>

<panel name=dfdxmp4>Library Inventory
  <topinst>To add a book to the inventory, complete the fields below,
  then press Enter.
  <area>
    <dtacol pmtwidth=15>
    <dtafld datavar=title usage=in entwidth=50>Title
    <dtafld datavar=author usage=in entwidth=20 deswidth=30>Author
      <dtafldd>Last name, First name, M.I.
    <dtafld datavar=publish entwidth=20>Publisher
    <dtafld datavar=pages usage=in entwidth=5 deswidth=15>
            Total number of pages
      <dtafldd>(1 - 99999)
    </dtacol>
  </area>
</panel>
```

Data Field Help

ISPF allows you to provide help on a data field using the HELP attribute on the DTAFLD tag. If you specify the name of a help panel or message for a data field, ISPF knows which help information to display when the user selects help on the

data field. If you do not specify help for a data field, the extended help panel (specified with the HELP attribute of the enclosing PANEL tag) is displayed.

The following example shows how to provide help for data fields.

```
<!doctype dm system>

<varclass name=titlcls type='char 50'>
<varclass name=bookcls type='char 20'>
<varclass name=pagecls type='numeric 5'>

<varlist>
  <vardcl name=title   varclass=titlcls>
  <vardcl name=author  varclass=bookcls>
  <vardcl name=publish varclass=bookcls>
  <vardcl name=pages   varclass=pagecls>
</varlist>

<panel name=dfdxmp5>Library Inventory
  <topinst>To add a book to the inventory, complete the fields below,
  then press Enter.
  <area>
    <dtacol pmtwidth=15>
    <dtafld datavar=title help=hlptitl entwidth=50>Title
    <dtafld datavar=author help=hlpauth entwidth=20 deswidth=30>Author
      <dtaflddd>Last name, First name, M.I.
    <dtafld datavar=publish help=hlppubl entwidth=20>Publisher
    <dtafld datavar=pages help=hlppage entwidth=5 deswidth=15>
           Total number of pages
      <dtaflddd>(1 - 99999)
    </dtacol>
   </area>
</panel>
```

Other Data Field Attributes

There are several other attributes you can specify to tailor a data field to meet the requirements of your application. See "DTAFLD (Data Field)" on page 312 for more information. The following list describes each of the remaining DTAFLD attributes and what you can do with them:

REQUIRED
This attribute allows you to indicate if the data field requires input. When you assign a value of YES to this attribute, the user must enter data into the field before ISPF accepts the panel as valid. The default REQUIRED value is NO. This attribute is only valid for data fields defined as input-only or as input/output.

MSG
This attribute identifies the message that should be displayed when the user does not enter any data into an input-required data field. If you do not specify this attribute, ISPF displays a default message. This attribute is valid only if REQUIRED=YES.

"Chapter 7. Messages" on page 159 tells you how to define application messages.

ALIGN
This attribute allows you to align the variable data within the data field. The default value for ALIGN is *start*, which aligns the data from the left side of the data field. You can also center the data within the field with the *center* value, or justify the data from the right side of the field with the *end* value.

AUTOTAB
This attribute provides automatic cursor movement between data fields. If you specify AUTOTAB=YES for a data field, the cursor automatically moves to the next field that is capable of input. If no other field capable of input exists on the panel, the cursor returns to the beginning of the data field.

DISPLAY
The value you assign to this attribute, either *yes* (the default) or *no*, determines if the data appears on the screen when the user enters it. One way to use DISPLAY=NO is for defining a password.

VARCLASS
This attribute allows you to override the variable class that is specified on the variable declaration (VARDCL) for the data field's data variable (DATAVAR). See "Chapter 4. Variables and Variable Classes" on page 61 for a description of variables and variable classes.

FLDSPACE
This attribute specifies the space reserved for the data-entry field. When the FLDSPACE value is larger than the entry width plus any attributes, blanks are added following the data-entry field. This provides spacing before DTAFLDD tag descriptions.

NOENDATTR
This attribute specifies that no ending attribute character will be placed after the data field. NOENDATTR is valid only when WINDOW=NO is specified or when data fields are being formatted within a horizontal region.

PAD
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

PADC
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

OUTLINE
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

PMTFMT
This attribute controls the generation of prompt leader characters. The default is to create CUA leader dots.

PSVAR
This attribute provides the name of a variable that is to be set when a DTAFLD is clicked on for point-and-shoot selection.

PSVAL
This attribute provides the value to be placed in the field specified by the PSVAR attribute.

PAS
This attribute can be used to provide a variable name that will contain the value ON to enable point-and-shoot for this data field, or OFF to disable point-and-shoot. When PSVAR and PSVAL have been specified without the PAS attribute, the point-and-shoot field will be automatically enabled.

CSRGRP
The CSRGRP attribute, in combination with the PAS attribute, is used to specify a cursor group for GUI mode operation.

EXPAND
The EXPAND attribute, used in combination with EXPAND=xy on the PANEL definition, causes the expand characters to be added to the DTAFLD definition, effectively allowing ENTWIDTH to expand.

FLDWIDTH
The FLDWIDTH attribute, used in combination with WINDOW=NO on the PANEL definition, provides the width of a data field that spans multiple lines.

ATTRCHANGE
The ATTRCHANGE attribute specifies that, if required, an additional)ATTR section entry (which can apply to multiple fields) be created instead of a unique ".ATTR" override entry for the current field.

INIT
The INIT attribute provides an initial value for the data field.

DBALIGN
The DBALIGN attribute is used only for DBCS language conversion when PMTLOC=ABOVE to align the prompt text with the data field.

DEPTH
This attribute defines the depth reserved for the field. When the panel is displayed in GUI mode, a field specified as point-and-shoot results in a push button displayed with the specified DEPTH.

IMAPNAME
This attribute specifies the name of an image to be placed on the point-and-shoot push button when it is displayed in GUI mode.

IMAPNAMEP
This attribute specifies the name of an image to be placed on the point-and-shoot push button after it has been pushed when it is displayed in GUI mode.

PLACE
This attribute specifies the position of the image relative to the text within the point-and-shoot push button.

PMTSKIP
This attribute, used during horizontal field formatting of input fields, specifies that the cursor should move past the prompt text to the input field.

DESSKIP
This attribute, used during horizontal field formatting of input fields, specifies that the cursor should move past the description text to the next input field.

FLDTYPE
This attribute specifies whether CUA or traditional ISPF attribute definitions are used.

COLOR
When FLDTYPE=ISPF, this attribute specifies the color of the field.

INTENS
When FLDTYPE=ISPF, this attribute specifies the intensity of the field.

HILITE
When FLDTYPE=ISPF, this attribute specifies the highlighting for the field.

ATTRCHAR
This attribute provides a user selected panel attribute for the data field.

CAPS
This attribute specifies whether the field is displayed in uppercase characters.

NOJUMP
This attribute specifies that the JUMP function is disabled for the data field.

AUTOTYPE
This attribute specifies whether ISPF panel logic is added to support the AUTOTYPE function.

AUTOVOL
This attribute specifies an associated volume name when AUTOTYPE = DSN.

AUTODMEM
This attribute specifies whether a member name is part of the data set name when AUTOTYPE = DSN.

VARDCL
This attribute specifies whether the field name is validated to the panel variables specified with the VARDCL tag.

Defining Selection Fields

Selection fields allow the user to select from a group of choices on an application panel. You can specify if only one choice can be selected from a selection field, or if multiple choices are allowed.

In either case, you use the same DTL tags to define a selection field. The SELFLD (selection field) tag and its required end tag define a selection field. The CHOICE (selection choice) tag defines a choice within a selection field. You code the CHOICE tags between the SELFLD start and end tags, like this:

```
<selfld>
  <choice>
  <choice>
  <choice>
</selfld>
```

Each CHOICE tag defines a choice within the selection field.

Like data fields, selection fields support field prompts, which can be placed in front of or above the selection field. Field prompts are described in "Field Prompts" on page 83.

To define the selection field type use the TYPE attribute of the SELFLD tag. The values you can assign to TYPE are:

SINGLE Specifies the selection field as being a single-choice field. Choices in a single-choice selection field appear in a list with an entry field preceding the first choice in the list. The conversion utility prefixes the text of each choice with a number, so the selection field choices are numbered sequentially. Users indicate choice selection by typing the number of the choice they want in the entry field.

MULTI Specifies the selection field as being a multiple-choice field. Choices in a multiple-choice selection field appear in a list with a single-character entry field preceding each choice. Users indicate choice selection by typing any nonblank character in the entry fields.

MENU Specifies the selection field as being a menu-choice field. Choices in

a menu-choice selection field are similar to those in a single-choice selection field. TYPE=MENU is valid only when the MENU keyword has been specified on the PANEL tag.

MODEL Specifies the selection field as being a model-choice field. Choices in a model-choice selection field are similar to those in a menu-choice selection field. TYPE=MODEL is valid only when the MENU keyword has been specified on the PANEL tag.

TUTOR Specifies the selection field as being a tutor-choice field. Choices in a tutor-choice selection field are similar to those in a menu-choice selection field. TYPE=TUTOR is valid only when the MENU keyword has been specified on the PANEL tag.

The CHOICE tag has two attributes associated with it that are important when defining a selection field: CHECKVAR and MATCH. The CHECKVAR and MATCH attributes are used to preselect choices in the selection field. The CHECKVAR attribute can also communicate to the application which selections were made by the user.

The value specified on the CHECKVAR attribute is the name of a dialog variable that is defined by the application. Both the application and ISPF can set the check variable. The following sections describe how the CHECKVAR and MATCH attributes are used for each type of selection field.

Single-Choice Fields

Use a single-choice selection field when you have a fixed set of choices that are mutually exclusive. That is, the user can select only one of the choices by typing the choice number in the entry field. You can specify the preselected choice in a single-choice selection field so that one item is already selected when the panel is displayed. The user can either leave the preselected choice or enter a different choice number.

To preselect choices in a single-choice selection field, and to find out which choice was selected by the user, you should specify the CHECKVAR and MATCH attributes for each CHOICE tag. For a single-choice field, all of the enclosed choices should refer to the same check variable, but they should have unique MATCH values. The following markup shows how this is coded.

```
<!doctype dm system>

<varclass name=daycls type ='char 1'>

<varlist>
  <vardcl name=day varclass=daycls>
  <vardcl name=choice varclass=daycls>
</varlist>

<panel name=singsel>Schedule Appointments
  <topinst>Choose the most convenient day for your appointment,
           then press Enter.
  <area>
    <selfld name=choice selwidth=30 pmtwidth=9>Weekdays:
      <choice checkvar=day  match=M>Monday
      <choice checkvar=day  match=T>Tuesday
      <choice checkvar=day  match=W>Wednesday
      <choice checkvar=day  match=H>Thursday
      <choice checkvar=day  match=F>Friday
    </selfld>
  </area>
</panel>
```

To preselect a certain choice, set the check variable, *day*, to the match value for that choice. Assume that the check variable, *day*, is set to *M* before the panel is displayed. When the panel is displayed, the choice, **Monday**, is selected as shown in Figure 37.

Figure 37. Single-Choice Selection Field

If the user decides that another day is more convenient, another choice might be selected. This causes the check variable to be updated with the match value of the newly selected choice. For example, if the user selects **Friday** (by typing "5" in the entry field), the check variable, *day*, will contain "F" when control is returned to the application.

Note: The TYPE attribute does not have to be specified on a single-choice selection field because TYPE=SINGLE is the default. However, you must specify the NAME attribute for single-choice selection fields.

Multiple-Choice Fields

Use a multiple-choice selection field when you have several choices for the user, but they are not mutually exclusive. Each choice acts independently as a toggle, and selecting one of the choices does not affect any of the other choices in the selection field.

To preselect choices in a multiple-choice selection field, and to find out which choices were selected by the user, specify the CHECKVAR, MATCH, and NOMATCH attributes for each CHOICE tag.

On a multiple-choice selection field, define a unique check variable for each enclosed CHOICE. You can let the MATCH value default to 1, or specify the MATCH attribute with a value of your choice. Also, you can let the NOMATCH value default to 0, or specify the NOMATCH attribute with a value of your choice. Here's how a multiple-choice selection field is coded:

```
<!doctype dm system>

<varclass name=sampcls type ='char 1'>

<varlist>
  <vardcl name=dry varclass=sampcls>
  <vardcl name=cut varclass=sampcls>
  <vardcl name=per varclass=sampcls>
  <vardcl name=fac varclass=sampcls>
  <vardcl name=man varclass=sampcls>
  <vardcl name=ped varclass=sampcls>
  <vardcl name=ch1 varclass=sampcls>
  <vardcl name=ch2 varclass=sampcls>
  <vardcl name=ch3 varclass=sampcls>
  <vardcl name=ch4 varclass=sampcls>
  <vardcl name=ch5 varclass=sampcls>
  <vardcl name=ch6 varclass=sampcls>
</varlist>

<panel name=multsel>Schedule Appointments
  <area>
    <dtacol pmtwidth=45 selwidth=76>
    <selfld type=multi>Choose the services needed, then press Enter.
      <choice name=ch1 checkvar=dry>Dry haircut
      <choice name=ch2 checkvar=cut>Shampoo, haircut, and style
      <choice name=ch3 checkvar=per>Permanent or body wave
      <choice name=ch4 checkvar=fac>Facial
      <choice name=ch5 checkvar=man>Manicure
      <choice name=ch6 checkvar=ped>Pedicure
    </selfld>
    </dtacol>
  </area>
</panel>
```

You specify preselected choices for a multiple-choice selection field just as you would for a single-choice selection field. Set the check variable for the preselected choices to the match values (or the default value of 1) for those choices. When a choice is preselected, a slash (/) is displayed in the entry field preceding the choice.

When the user types a value in an entry field in a multiple-choice selection field, ISPF toggles the choice as follows:

- If the choice is already selected and the user enters a blank in the entry field, ISPF deselects the choice and sets the check variable to the NOMATCH value for the choice, or to 0 if the NOMATCH attribute is not specified.
- If the choice is not selected and the user types a nonblank character in the entry field, ISPF selects the choice and sets the check variable to the MATCH value for the choice, or to 1 if the MATCH attribute is not specified. If the choice is not selected, ISPF sets the check variable to the NOMATCH value for the choice, or to 0 if the NOMATCH attribute is not specified.

In the preceding markup, the MATCH attribute was not specified, so the check variables toggle between 0 and 1 (the default MATCH and NOMATCH values) as the user selects and deselects items.

Because ISPF is setting the check variable, you should not use the SETVAR or the TOGVAR attributes of the ACTION tag to refer to the check variable.

Figure 38 shows how the multiple-choice selection field in the preceding markup appears with the choices **Facial** and **Pedicure** preselected.

Figure 38. Multiple-Choice Selection Field

Menu-Choice Fields

Use a menu-choice selection field to create an ISPF option menu. Menu-choice fields are similar to single-choice fields. That is, the user can select only one of the choices presented. The entry field for this type of selection field is the command line, which is formatted with the word *Option* instead of *Command*. As with single-choice selections, you can specify a preselected choice so that one item is already selected when the panel is displayed.

The CHOICE tag is followed by an ACTION tag which specifies the type of selection (PANEL, PGM, CMD, or EXIT), and other attributes required by the ISPF SELECT service.

When creating an option menu, the MENU keyword is required on the PANEL tag. The optional PRIME keyword causes the creation of a primary option menu. The SELFLD tag must specify TYPE=MENU. Depending on the panel being created, the SELFLD tag attributes ENTWIDTH, FCHOICE, and TRAIL, and the CHOICE tag attribute SELCHAR might be required. See “Chapter 13. Tag Reference” on page 211 for more information on the PANEL, SELFLD, CHOICE, and ACTION tags.

The following markup creates a sample option menu:

```
<!doctype dm system ()>
<!--  MENU selection panel example -->
<panel name=menusel1 menu>Sample Option Menu
  <topinst>Enter a selection choice

  <region indent=4>
    <selfld type=menu entwidth=1 selwidth=40>
      <choice checkvar=xtest1 match=a>Select Command
        <action run=tstch1 type=cmd parm='1234'
```

```
                newappl=aaaa passlib newpool suspend
                lang=crex nocheck mode=fscr>
      <choice checkvar=xtest1 match=b>Select Panel
        <action run=tstch2 type=panel
                addpop newappl=aaaa passlib newpool suspend>
      <choice checkvar=xtest1 match=c>Select Program
        <action run=tstch3 type=pgm parm=abcd
                newappl=aaaa passlib newpool suspend
                nocheck mode=fscr>
      <choice checkvar=xtest1 match=x>Exit
        <action run=exit type=exit>
    </selfld>
  </region>

  <cmdarea>
</panel>
```

The resulting panel is:

```
                            Sample Option Menu

 Enter a selection choice

     1 Select Command
     2 Select Panel
     3 Select Program
     4 Exit

Option ===> ____________________________________________________________
```

Figure 39. Sample Option Menu

Model-Choice Fields

Use a model-choice selection field to create an ISPF edit model selection menu. Model-choice fields are similar to single-choice or menu-choice fields. That is, the user can select only one of the choices presented. The entry field for this type of selection field is the command line, which is formatted with the word *Option* instead of *Command*. As with single-choice or menu-choice selections, you can specify a preselected choice so that one item is already selected when the panel is displayed.

The CHOICE tag is followed by an ACTION tag which specifies the type of selection (PANEL, PGM, CMD, or EXIT), and other attributes required by the ISPF SELECT service.

When creating an edit model menu, the MENU keyword is required on the PANEL tag. The SELFLD tag must specify TYPE=MODEL. Depending on the panel being

created, the SELFLD tag attributes ENTWIDTH, FCHOICE, and TRAIL, and the CHOICE tag attributes SELCHAR, HIDEX, and TRUNC might be required. See "Chapter 13. Tag Reference" on page 211 for more information on the PANEL, SELFLD, CHOICE, and ACTION tags.

Tutor-Choice Fields

Use a tutor-choice selection field to create an ISPF tutorial selection menu. Tutor-choice fields are similar to menu-choice fields. That is, the user can select only one of the choices presented. The entry field for this type of selection field is the command line, which is formatted with the word *Option* instead of *Command*. As with menu-choice selections, you can specify a preselected choice so that one item is already selected when the panel is displayed.

The CHOICE tag is followed by an ACTION tag that must specify the type of selection as PANEL, and other attributes required by the ISPF SELECT service.

When creating a tutorial menu, the MENU keyword is required on the PANEL tag. The SELFLD tag must specify TYPE=TUTOR. Depending on the panel being created, the SELFLD tag attributes ENTWIDTH and FCHOICE, and the CHOICE tag attribute SELCHAR might be required. See "Chapter 13. Tag Reference" on page 211 for more information on the PANEL, SELFLD, CHOICE, and ACTION tags.

Selection Field Help

ISPF enables you to provide help on selection fields. For single-choice selection fields, you specify the name of a help panel or message for the selection field with the HELP attribute of the SELFLD tag. For multiple-choice selection fields, you specify the name of a help panel or message for each of the choices in the selection field with the HELP attribute of the CHOICE tags. For menu-choice, model-choice, or tutor-choice fields, the selection field is the command line. The name of the help panel or message must be provided on the CMDAREA tag. If you specify help for a single-choice selection field, a menu-choice selection field, or for choices in a multiple-choice selection field, ISPF displays that help information when the user requests help and the cursor is on that panel element. If there is no help defined, the extended help panel is displayed.

The following example shows how to code a help panel for a single-choice selection field.

```
<selfld name=choice help=dayhelp>Weekdays:
  <choice checkvar=day match=M>Monday
  <choice checkvar=day match=T>Tuesday
  <choice checkvar=day match=W>Wednesday
  <choice checkvar=day match=H>Thursday
  <choice checkvar=day match=F>Friday
</selfld>
```

This example shows how to code help panels for choices in a multiple-choice selection field.

```
<selfld type=multi>Choose the services needed:
  <choice name=ch1 help=dryhlp>Dry haircut
  <choice name=ch2 help=cuthlp>Shampoo, haircut, and style
  <choice name=ch3 help=permhlp>Permanent or body wave
  <choice name=ch4 help=facehlp>Facial
  <choice name=ch5 help=manihlp>Manicure
  <choice name=ch6 help=pedihlp>Pedicure
</selfld>
```

Selection Width

The SELWIDTH attribute of the SELFLD tag should be used to define the amount of space taken up by the *choice-description-text* of each CHOICE tag. This attribute is used to control the formatting of panels defined with horizontal regions. If you do not specify a SELWIDTH value, the conversion utility reserves the remaining available formatting width for the text.

When specifying an explicit SELWIDTH value, you must take into consideration the components of the selection field, as well as the *choice-description-text*. The conversion utility reserves a number of positions on the lines that selection field choices appear on for the entry fields, 3270 attributes, and, in the case of single-choice, menu-choice, model-choice, and tutor-choice selection fields, the choice prefixes. See page 458 for a discussion of the amount of space reserved for each choice type.

These reserved positions must be added to the length of the *choice-description-text* in the SELWIDTH value you specify. For example, the following markup contains two selection fields, one single-choice and one multiple-choice, within a horizontal region. To format the selection fields properly, ensure that the SELWIDTH values you specify are adequate for the reserved positions and the *choice-description-text*. The largest *choice-description-text* in the first selection field is 9 characters, which, when combined with the 10 reserved positions in the field, means you must specify a SELWIDTH value of at least 19. The largest *choice-description-text* in the second selection field is 27 characters, which, when combined with the 5 reserved positions in the field, means you must specify a SELWIDTH value of at least 32.

See page 458 for more information aboout SELWIDTH.

```
<!doctype dm system>
<varclass name=char1 type='char 1'>
<varclass name=char2 type='char 2'>

<varlist>
  <vardcl name=person varclass=char2>
  <vardcl name=ch1 varclass=char1>
  <vardcl name=ch2 varclass=char1>
  <vardcl name=ch3 varclass=char1>
  <vardcl name=ch4 varclass=char1>
  <vardcl name=ch5 varclass=char1>
  <vardcl name=ch6 varclass=char1>
</varlist>

<panel name=servsel>Service Selections
  <topinst>Select the stylist and services you want, then press Enter.
  <area>
    <region dir=horiz>
    <selfld name=person selwidth=19 pmtwidth=15>Stylist
      <choice checkvar=stylst match=1>Cecilia
      <choice checkvar=stylst match=2>Dana
      <choice checkvar=stylst match=3>Laurel
      <choice checkvar=stylst match=4>Pierce
      <choice checkvar=stylst match=5>Stephenie
    </selfld>
    <divider>
    <selfld type=multi selwidth=32 pmtwidth=15>Services
      <choice name=ch1 checkvar=dry>Dry haircut
      <choice name=ch2 checkvar=cut>Shampoo, haircut, and style
      <choice name=ch3 checkvar=per>Permanent or body wave
      <choice name=ch4 checkvar=fac>Facial
      <choice name=ch5 checkvar=man>Manicure
      <choice name=ch6 checkvar=ped>Pedicure
```

```
    </selfld>
    </region>
  </area>
</panel>
```

Here's the formatted result:

```
                          Service Selections

Select the stylist and services you want, then press Enter.

Stylist               Services
__  1.  Cecilia       _  Dry haircut
    2.  Dana          _  Shampoo, haircut, and style
    3.  Laurel        _  Permanent or body wave
    4.  Pierce        _  Facial
    5.  Stephenie     _  Manicure
                      _  Pedicure
```

Figure 40. Selection Field SELWIDTH Attribute

Other Selection Field Attributes

There are several other attributes you can specify to tailor a selection field to meet the requirements of your application. See "SELFLD (Selection Field)" on page 455 for more information. The following list describes each of the remaining SELFLD attributes and what you can do with them:

ENTWIDTH
This attribute controls the entry width for single-choice, menu-choice, model-choice, and tutor-choice selections.

REQUIRED
This attribute allows you to indicate if the single-choice selection field requires input. When you assign a value of YES to this attribute, the user must enter data into the field before ISPF accepts the panel as valid. The default REQUIRED value is NO.

MSG
This attribute identifies the message that should be displayed when the user does not enter any data into the selection field. If you do not specify this attribute, ISPF displays a default message. This attribute is valid only if REQUIRED=YES.

"Chapter 7. Messages" on page 159 tells you how to define application messages.

FCHOICE
This attribute controls the first choice number for single-choice, menu-choice, model-choice, and tutor-choice selections. The value can be either 0 or 1.

AUTOTAB
This attribute provides automatic cursor movement between fields. If you specify AUTOTAB=YES for a selection field, the cursor automatically moves to the next field that is capable of input. If no other field capable of input exists on the panel, the cursor returns to the selection field.

DEPTH
This attribute specifies that the selection list is to be formatted as a scrollable area. A list formatted into multiple columns (see CHOICECOLS below) is formatted as multiple scrollable areas.

EXTEND
This attribute is valid only when DEPTH has been specified and specifies that the scrollable area is to be expanded at run-time to the size of the logical screen.

TRAIL
This attribute is used with menu-choice selections to specify the name of one or more variables that applications use to obtain TRAIL information created by option menu selection processing.

CHOICECOLS
This attribute is used to specify the number of columns to create for the selection list. When multiple columns are requested, the number of choices placed in each column is obtained from the CHOICEDEPTH attribute.

CHOICEDEPTH
This attribute specifies the number of choices to be formatted into each column of choices. If more choice entries are specified than can be formatted in the available number of columns specified by the CHOICECOLS attribute, the remaining choice entries are placed in the rightmost (or only) available column for the current SELFLD tag.

CWIDTHS
This attribute specifies the number of bytes to be allocated for each column of CHOICE entries. The 'w1 w2...wn' notation provides the number of bytes for each column. You may use an asterisk or a number combined with an asterisk to specify a proportional allocation of column space.

PAD
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

PADC
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

OUTLINE
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

SELMSG
This attribute specifies the message that is displayed when an invalid single-choice entry is selected.

SELMSGU
This attribute specifies the message that is displayed when an unavailable single-choice entry is selected.

INIT
This attribute controls the single-choice and multiple-choice selection field variables initialization in the panel)INIT section.

VERIFY
This attribute controls the single-choice verification and menu-choice, model-choice, or tutor-choice selection logic generation in the panel)PROC section.

REFRESH
This attribute controls the creation of the REFRESH statement in the)REINIT section for multi-choice selection variables.

SELFMT
This attribute controls the placement of the choice selection character(s) within the width specified by ENTWIDTH.

CHKBOX
This attribute controls the display of multiple-choice fields as check boxes when operating in GUI mode.

ZGUI
This attribute controls the creation of the VGET (ZGUI) statement created as part of the)INIT section for multiple-choice selection definitions using the "&multipmt" built-in ENTITY.

CSRGRP
This attribute, in combination with CHKBOX=YES, provides a cursor group identification for multi-choice selections.

TSIZE
This attribute provides the number of bytes to indent multiple lines of CHOICE text.

LISTTYPE
This attribute controls the display of single-choice selection lists when operating in GUI mode.

LISTREF
This attribute provides the name of the)LIST section for list boxes, drop-down lists, and combination boxes.

LISTDEPTH
This attribute specifies the display depth for list boxes, drop-down lists, and combination boxes.

DBALIGN
This attribute, used for DBCS fields when PMTLOC=ABOVE, specifies alignment of the prompt text with the selection input field.

NOSEL
This attribute provides a value to be placed in the CHECKVAR variable (specified by the CHOICE tag), when no selection is made from a single-choice selection list.

SELDEFAULT
This attribute specifies a default choice selection for a single-choice selection list.

PMTSKIP
This attribute, used during horizontal field formatting, specifies that the cursor should move past the prompt text to the input field.

FLDTYPE
This attribute specifies whether CUA or traditional ISPF attribute definitions are used.

COLOR
When FLDTYPE=ISPF, this attribute specifies the color of the the field.

INTENS
When FLDTYPE=ISPF, this attribute specifies the intensity of the the field.

HILITE
When FLDTYPE=ISPF, this attribute specifies the highlighting of the field.

SELCHECK
This attribute is used with menu-choice selection to specify that panel logic be included in selection processing to check for selection choices that are not valid.

Data Columns

The DTACOL (data column) tag can be used to define values for data fields and selection fields that are coded within the data column. If you have a group of data fields and selection fields on the same application panel, the DTACOL tag is a convenient short-cut for ensuring alignment of the fields.

The DTACOL tag has the following attributes:

PMTWIDTH Applies to data fields and selection fields

ENTWIDTH Applies to data fields only

DESWIDTH Applies to data fields only

SELWIDTH Applies to selection fields only

FLDSPACE Applies to data fields only

PAD Applies to data fields only

PADC Applies to data fields only

OUTLINE Applies to data fields only

PMTFMT Applies to data fields only

AUTOTAB Applies to data fields only

ATTRCHANGE
Applies to data fields only

PMTLOC Applies to data fields only

DBALIGN Applies to data fields only

VARCLASS Applies to data fields only

REQUIRED Applies to data fields only

CAPS Applies to data fields only

These attributes serve the same purposes in DTACOL definitions as they do in CHOFLD, DTAFLD, and SELFLD definitions. The only difference is that when you use them with a DTACOL tag, they define those values for all of the data fields and selection fields coded between the DTACOL start and end tags.

The following markup uses a data column to define a prompt width, entry width, and description width for the data fields and the selection field coded within the data column. Because we want to limit the entry width of the **State** and **Zip code** fields, we defined ENTWIDTH values in the DTAFLD definitions for these fields that override the DTACOL ENTWIDTH value.

```
<!doctype dm system>

<varclass name=sampcls type ='char 30'>
<varclass name=statcls type ='char 2'>
<varclass name=zipcls  type ='char 5'>
<varclass name=char1cls  type ='char 1'>

<varlist>
  <vardcl name=name varclass=sampcls>
  <vardcl name=addr varclass=sampcls>
  <vardcl name=city varclass=sampcls>
  <vardcl name=stat varclass=statcls>
  <vardcl name=day  varclass=char1cls>
  <vardcl name=zipc varclass=zipcls>
 </varlist>

<panel name=dcolxmp>Schedule Appointments
  <topinst>Enter your name and address and
  choose the most convenient day for your appointment.
  <area>
    <dtacol pmtwidth=12 entwidth=30 deswidth=29 selwidth=30>
      <dtafld datavar=name>Name
        <dtafldd>Last, First, M.I.
      <dtafld datavar=addr>Address
        <dtafldd>If it applies, include apartment number
      <dtafld datavar=city>City
      <dtafld datavar=stat entwidth=2>State
        <dtafldd>Use 2-character abbreviation
      <dtafld datavar=zipc entwidth=5>Zip code
      <divider type=solid gutter=3>
      <selfld name=day pmtloc=before>Weekdays
        <choice>Monday
        <choice>Tuesday
        <choice>Wednesday
        <choice>Thursday
        <choice>Friday
      </selfld>
    </dtacol>
  </area>
</panel>
```

Here's how the panel formats:

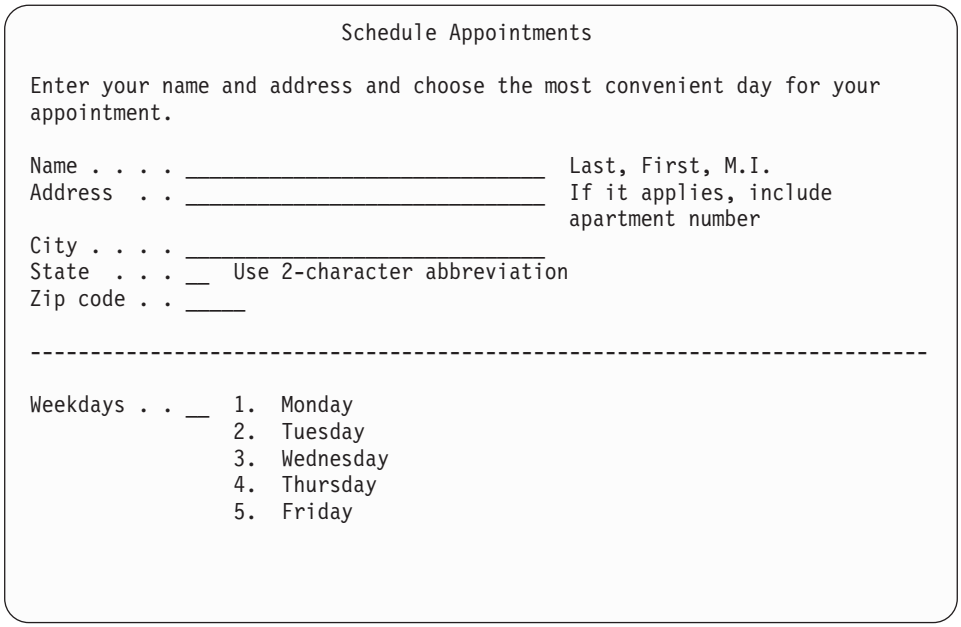


Figure 41. Data Column

Defining List Fields

A list field is used to display ISPF table data in column format, and to allow the user to enter data in the column rows. The list field supports vertical scrolling if all of the data in the list field is not visible.

If you define a list field in a panel, the ISPF application program must use the TBDISPL service to display the panel.

The tags you use to define a list field are:

LSTFLD To define the list field. A matching end tag is required.

LSTGRP To define column group headings. A matching end tag is required.

LSTCOL To define a column within a list field. You code a LSTCOL tag for each column of data in the list field.

LSTVAR To define a variable model line.

A list field can contain one or more columns of data, where each column can be input-only, output-only, or input/output, as defined by the USAGE attribute on the LSTCOL tag. These are the values you can specify on the USAGE attribute:

IN Defines an input-only list column. An input-only column is underscore-filled when it is initially displayed, unless the data is right-justified, and the user can enter data into any of the rows in the input column.

OUT Defines an output-only list column. When the panel is initially displayed, output-only columns display the value of the ISPF table variable associated with the list column. The user cannot interact with an output-only list column.

BOTH Defines an input/output list column. Input/output list columns display the value of the ISPF table variable associated with the list column when the panel is initially displayed, as well as allowing the user to enter data into any of the rows in the column. BOTH is the default value for the USAGE attribute.

The data that is associated with each list column is specified on the DATAVAR attribute of the LSTCOL tag. Like all variables used on the panel, the data variable should be declared using the VARDCL tag.

The conversion utility builds a model section into the converted application panel. The model section begins with a)MODEL header statement, which includes the variables named by the DATAVAR attributes of each of the LSTCOL tags defined within the LSTFLD.

Application panels defined using the LSTFLD tag must be displayed using the ISPF TBDISPL service. You can specify the optional ROWS=SCAN attribute on the LSTFLD tag to indicate that only those rows meeting the criteria established by a previous TBSARG service are to be displayed.

You can define a column heading for any of the list columns in the list field by specifying the column heading text as the tag text on the LSTCOL tag. You can specify the optional DIV attribute on the LSTFLD tag to create a divider line between the display of table rows. The column headings do not scroll when the list field is scrolled.

A scroll amount field can be placed at the right end of the command line by specifying the SCROLLVAR attribute on the LSTFLD tag. Field level help for the SCROLLVAR field is specified using the SCRVHELP attribute. The scroll amount field is displayed in uppercase characters when the SCRCAPS=ON attribute is specified.

This panel shows a list field with six columns. The first column is output-only, and the remaining columns are input/output.

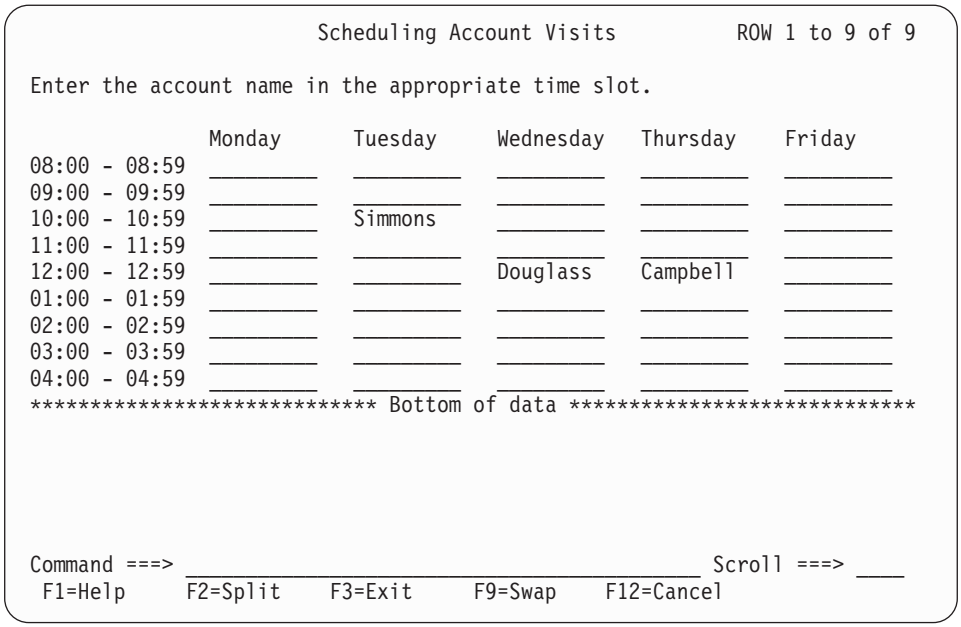


Figure 42. List Field

Here's the markup we used to create the panel:

```
<!doctype dm system>

<varclass name=timecls type='char 13'>
<varclass name=vc1     type ='char 9'>

<varlist>
  <vardcl name=timecol varclass=timecls>
  <vardcl name=moncol  varclass=vc1>
  <vardcl name=tuecol  varclass=vc1>
  <vardcl name=wedcol  varclass=vc1>
  <vardcl name=thrcol  varclass=vc1>
  <vardcl name=fricol  varclass=vc1>
</varlist>

<panel name=lstfld2>Scheduling Account Visits
 <topinst>Enter the account name in the appropriate time slot.
 <area>
  <lstfld scrollvar=scrlamt scrvhelp=scrhelp>
    <lstcol datavar=timecol usage=out colwidth=13>
    <lstcol datavar=moncol colwidth=9>Monday
    <lstcol datavar=tuecol colwidth=9>Tuesday
    <lstcol datavar=wedcol colwidth=9>Wednesday
    <lstcol datavar=thrcol colwidth=9>Thursday
    <lstcol datavar=fricol colwidth=9>Friday
  </LSTFLD>
</area>
<cmdarea>
</panel>
```

List Group Headings

You can define additional headings for the columns in a list field using the LSTGRP (list group) tag and its matching end tag. You can define a list group for a single list column or for multiple list columns. You nest the list columns you want to provide additional heading text for within the LSTGRP definition.

At least one field from the first line of the model set must be included within a LSTGRP definition.

The HEADLINE attribute of the LSTGRP tag allows you to place dashes in the list group heading. This is handy for list groups that span across several list columns. Specify HEADLINE=YES to produce a dashed list group heading.

The ALIGN attribute of the LSTGRP tag allows you to control the format position of the list group heading. The default value is CENTER. The heading can be left- or right-justified by specifying the values START or END, respectively.

For example, we added a LSTGRP definition to the list field shown earlier.

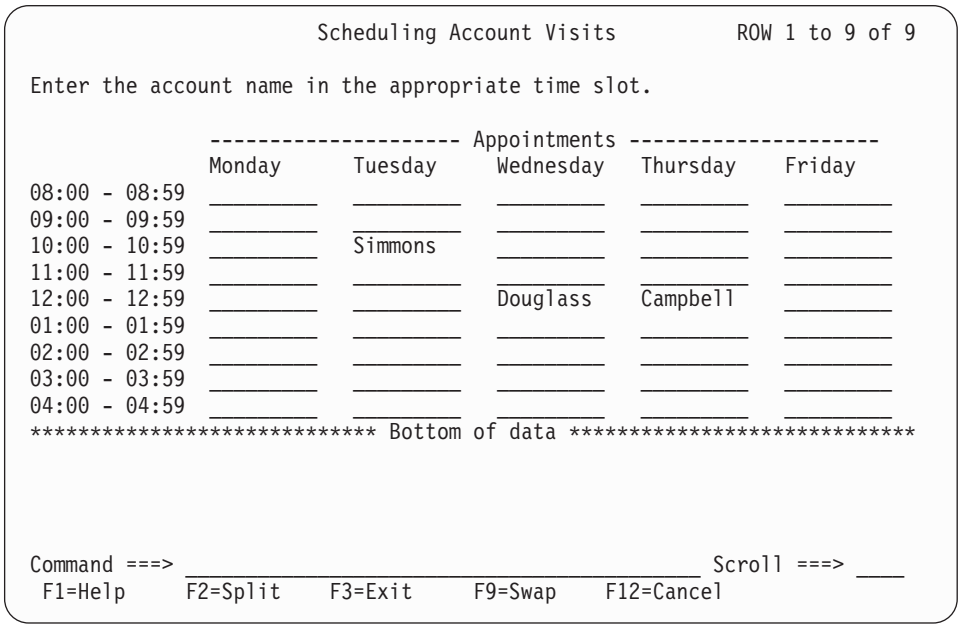


Figure 43. List Group

The text of the list group, **Appointments** is centered within the dashes. Here's how we coded the list group:

```
<!doctype dm system>

<varclass name=timecls type='char 13'>
<varclass name=vc1     type ='char 9'>

<varlist>
  <vardcl name=timecol varclass=timecls>
  <vardcl name=moncol  varclass=vc1>
  <vardcl name=tuecol  varclass=vc1>
  <vardcl name=wedcol  varclass=vc1>
  <vardcl name=thrcol  varclass=vc1>
  <vardcl name=fricol  varclass=vc1>
</varlist>

<panel name=lstgrp2>Scheduling Account Visits
 <topinst>Enter the account name in the appropriate time slot.
 <area>
  <lstfld scrollvar=scrlamt scrvhelp=scrhelp>
    <lstcol datavar=timecol usage=out colwidth=13>
```

```
    <lstgrp headline=yes>Appointments
      <lstcol datavar=moncol colwidth=9>Monday
      <lstcol datavar=tuecol colwidth=9>Tuesday
      <lstcol datavar=wedcol colwidth=9>Wednesday
      <lstcol datavar=thrcol colwidth=9>Thursday
      <lstcol datavar=fricol colwidth=9>Friday
    </lstgrp>
  </lstfld>
</area>
<cmdarea>
</panel>
```

List Column Width

You can use the COLWIDTH attribute of the LSTCOL tag to determine the data width to be used by the column. If you do not specify this attribute, the data width and column formatting width are determined by the actual length of the *column-heading* If the width of the *column-heading* text is greater than the COLWIDTH, it is used as the column formatting width.

The minimum width value is 1 and the maximum is the remaining available panel (or region) width. If the *column-heading* and the COLWIDTH attribute are omitted, the data width and column formatting width are determined by the TYPE value of the associated VARCLASS. If a VARCLASS TYPE value is not available, the size of the column variable name (specified by the DATAVAR attribute) determines the width.

You should code the COLWIDTH attribute with a value equal to the length of the table data variable.

Other List Column Attributes

There are several other attributes that can be used in the LSTCOL tag. Many of these attributes are the same as attributes on the DTAFLD tag. The following list describes these LSTCOL attributes and how they are used:

ALIGN
This attribute aligns the variable data within the list column. The default value for ALIGN is *start*, which aligns the data from the left side of the column. You can also center the data within the column with the *center* value, or justify the data to the right side of the column with the *end* value. The attribute value *end* is useful for right-justifying numbers within an output-only column, because numbers are typically right-aligned.

ATTRCHANGE
This attribute specifies that, if required, an additional)ATTR section entry (which can apply to multiple fields) be created instead of a unique ".ATTR" override entry for the current field.

AUTOTAB
This attribute specifies automatic tabbing. If you assign a value of YES to this attribute, the cursor automatically moves to the next field that is capable of user input when the user enters the last character in the current list column. The default value for AUTOTAB is NO. This attribute is only valid for list columns defined as input-only or as input/output.

CLEAR
This attribute specifies that the column is a table extension variable, which should be cleared before the row is displayed. Column names with the CLEAR attribute are identified by the CLEAR keyword on the)MODEL statement.

COLOR
When COLTYPE=ISPF, this attribute specifies the color for the column.

COLSPACE
The COLSPACE attribute specifies the total number of bytes for the column width, including the leading and trailing attributes, and the trailing blank for input fields. The use of the COLSPACE attribute causes column heading text longer than the COLSPACE value to be flowed into multiple lines.

COLTYPE
The COLTYPE attribute specifies the attribute type to be used for the column.

CSRGRP
This attribute, in combination with the PAS attribute, specifies a cursor group for GUI mode operation.

FORMAT
This attribute specifies how the data column and its column heading are formatted. If you do not specify this attribute, or if you specify the attribute value START, then the column formats as in ISPF Version 3.1 and ISPF Version 3.2.

HELP
This attribute specifies the help panel name to display when the user requests help on the list column.

HILITE
When COLTYPE=ISPF, this attribute specifies the highlighting for the column.

INTENS
When COLTYPE=ISPF, this attribute specifies the intensity for the column.

LINE
This attribute specifies the model line that contains the variable. You can specify lines 1–8.

MSG
This attribute identifies the message that should be displayed when the user does not enter any data into an input-required list column. If you do not specify this attribute, ISPF displays a default message. This attribute is valid only if REQUIRED=YES. “Chapter 7. Messages” on page 159 tells you how to define application messages.

NOENDATTR
This attribute specifies that no ending attribute character will be placed after the data column. NOENDATTR is ignored for the last data column on each model line. See “LSTCOL (List Column)” on page 372 for more information about the NOENDATTR attribute.

OUTLINE
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a “%”. See “LSTCOL (List Column)” on page 372 for more information about the OUTLINE attribute.

PAD
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a “%”. See “LSTCOL (List Column)” on page 372 for more information about the PAD attribute.

PADC
This attribute specifies the conditional padding character to be used for

initializing the field. You can define this attribute as a variable name preceded by a "%". See "LSTCOL (List Column)" on page 372 for more information about the PADC attribute.

PAS
This attribute is used to control the generation of the point-and-shoot indicator for table display panels. You can define this attribute as a variable name preceded by a "%".

POSITION
This attribute allows you to specify the starting position of the data column. The POSITION value must be greater than the end of the last formatted data column for that model line and less than the right panel margin. Column formatting for adding the data column and text takes place after the starting position has been established. See "LSTCOL (List Column)" on page 372 for more information.

REQUIRED
This attribute indicates if this column is required to have input for any modified row. For input-required columns (REQUIRED=YES), ISPF will not validate the panel unless the user has entered data into that column. If you do not specify this attribute, input is not required on the list column. This attribute is only valid for list columns defined as input-only or as input/output.

TEXT
This attribute specifies a short description of the data column. Text can be placed before or after the data column. See "LSTCOL (List Column)" on page 372 for more information.

TEXTLOC
This attribute specifies the location of the TEXT relative to the data column. Text can be placed on either side of the data column. See "LSTCOL (List Column)" on page 372 for more information.

TEXTFMT
This attribute specifies the format of the text within the length of the text area. The text can be left-justified, centered, or right-justified. See "LSTCOL (List Column)" on page 372 for more information.

TEXTLEN
This attribute specifies the amount of space to reserve for formatting the descriptive text. This helps you line up text on different model lines, and if the space reserved is longer than the descriptive text, TEXTLEN permits formatting within the reserved space with the TEXTFMT attribute. See "LSTCOL (List Column)" on page 372 for more information.

TEXTSKIP
This attribute specifies the cursor should move past the text to the next input field.

VARCLASS
This attribute allows you to override the variable class that is specified on the variable declaration (VARDCL) for the list column's data variable (DATAVAR). See "Chapter 4. Variables and Variable Classes" on page 61 for a description of variables and variable classes.

CAPS
This attribute specifies whether the data column is displayed in uppercase characters.

DISPLAY
This attribute specifies whether the data column is visible when the panel is displayed.

Defining Group Headings

The Group Header (GRPHDR) tag defines a group heading in the panel)BODY section.

The FORMAT attribute is used to control the type of text formatting. You can choose formatting similar to the LINES tag or the P tag. For example, if FORMAT=NONE, the text formats as if you used a LINES tag. However, if FORMAT=START, CENTER, or END, the text flows to multiple lines and is formatted at the right, center or left part of the space reserved for the group heading.

The following list provides a short description of the other available attributes:

WIDTH
This attribute specifies the number of columns reserved for the group heading. The default value is the remaining panel width.

FMTWIDTH
This attribute specifies the number of columns (of the WIDTH value) to use for formatting the group heading. The default is the WIDTH value. By specifying a FMTWIDTH that is less than the WIDTH value, the group heading text can be formatted on multiple lines.

INDENT
This attribute specifies the number of bytes that the group heading is to be indented.

HEADLINE
This attribute specifies whether dashes are added to span the width of the group heading not occupied by text.

DIV
This attribute specifies the type of divider line to be placed before and after the group heading text.

DIVLOC
This attribute specifies whether the divider is to be added before the group heading, after the group heading, or both before and after the group heading.

COMPACT
This attribute causes the group heading to format without a blank line before the group heading.

STRIP
This attribute causes leading and trailing blanks to be removed from the group heading text.

Defining Point-and-Shoot Fields

The Point-and-Shoot (PS) tag is used to identify a portion of panel)BODY section text to be used for point-and-shoot selection. When a point-and-shoot selection is made, a variable is set to a specified value before normal)PROC section processing. The PS tag attributes identify the variable name and the value associated with each point-and-shoot selection.

The PS tag requires a matching end tag to indicate the end of the point-and-shoot text.

Refer to the *ISPF Dialog Developer's Guide and Reference* for more information about point-and-shoot selection.

Chapter 6. Information Regions and Help Panels

Some of the information displayed on panels is *static*, or fixed text that the user does not interact with directly. This includes text such as top instructions and bottom instructions, prompt text, and data-field description text. DTL provides you with another method of defining static text for application panels using *information regions*.

Defining an information region on a panel allows you more flexibility for defining static text on a panel. The tags you use to define the text of information regions are much more versatile than the tags you use to define other types of static text, which means you can be more creative in the text you define.

In addition to using information regions on application panels, you must use them to define the text on help panels you define for your application. In this chapter, we tell you how to define information regions on application panels, and how to define help panels for your applications.

Defining an Information Region

Use the INFO tag and its required end tag to define an information region on a panel. You can code an information region within an AREA, HELP, PANEL, or REGION definition.

Here's an example of an INFO definition:

```
<panel name=infopan width=42 depth=16>Information

  <area>
    <info>
    </info>
  </area>
</panel>
```

The INFO tag has an optional WIDTH attribute that defines the width of the information region. If the value you assign the INFO WIDTH attribute is greater than the WIDTH available in the panel, the conversion utility will reset the value to the available width.

Note: You should code the WIDTH attribute if the information region is part of an application panel definition that uses horizontal region capability.

The INFO tag only defines an information region. It does not define the text of the information region. DTL provides you with a set of tags that define the text in information regions. These tags are:

- ATTENTION
- CAUTION
- DL (definition list)
- FIG (figure)
- Hn (heading)
- HP (highlighted phrase)
- LINES
- NOTE
- NOTEL (note list)
- NT (note)

© Copyright IBM Corp. 1989, 2001

- OL (ordered list)
- P (paragraph)
- PARML (parameter list)
- RP (reference phrase)
- PS (point-and-shoot)
- SL (simple list)
- UL (unordered list)
- WARNING
- XMP (example).

With the exception of HP, PS, and RP, these tags can be coded *only* within an INFO definition. The next section explains how to use each of these tags and some other tags that complement these tags within information regions.

Defining Basic Text

Paragraphs

The tag you use most often in information regions is the P (paragraph) tag. Use the P tag to arrange text as you would arrange a paragraph in your usual writing (to join one or more sentences related by their subject matter into a single block of text).

When the paragraph text formats for display, the text starts at the current margin and the words automatically wrap to fit within the margin. In addition, the conversion utility normally inserts a blank line before each paragraph.

The P tag has an optional attribute, COMPACT, which causes the blank line before the paragraph to be omitted. The P tag does not require a matching end tag.

We'll illustrate the use of the P tag with this example :

```
<!doctype dm system>
<panel name=infopan1 width=42>Information
  <area>
    <info width=40>
      <p>This is a paragraph.
      This sentence is also part of the paragraph.
    </info>
  </area>
</panel>
```

Notice that we coded the second sentence of the paragraph on a different line. It doesn't matter, because the conversion utility treats it as part of the same paragraph and formats it accordingly. That is, two blanks are automatically inserted between the sentences. Here's how the paragraph looks:

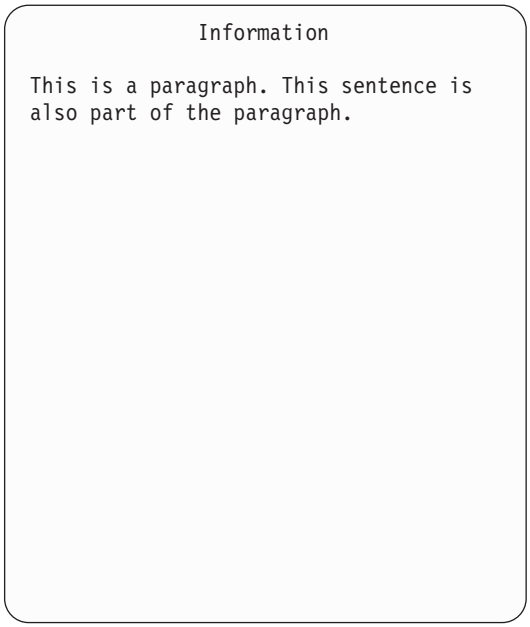


Figure 44. Paragraph

As you can see, the text of the paragraph is left-justified on the panel and the words automatically wrap to fit within the defined dimensions of the information region.

We'll add another paragraph to the panel to illustrate how two paragraphs format:

```
<!doctype dm system>
<panel name=infopan2 width=42>Information
  <area>
    <info width=40>
      <p>This is a paragraph.
      This sentence is also part of the paragraph.
      <p>Here is another paragraph.
      Paragraphs are useful for providing
      information on panels.
    </info>
  </area>
</panel>
```

Figure 45 on page 118 shows the result:

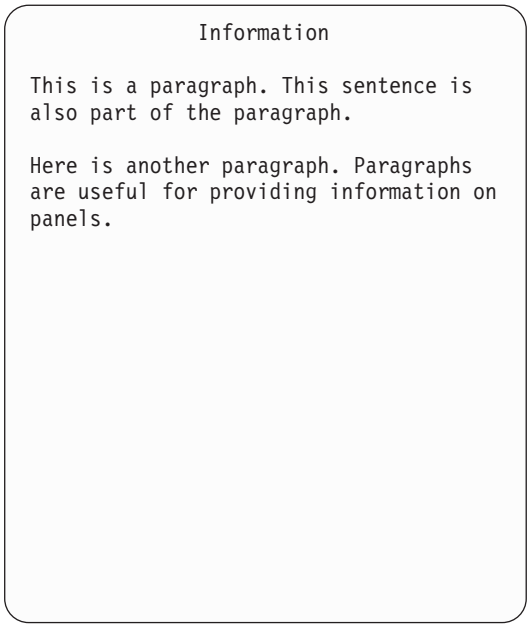


Figure 45. Multiple Paragraphs

In addition to the placement and wrapping of the text, the compiler separated the paragraphs with a blank line.

Headings

The Hn (heading) tag allows you to place headings in an information region. You use these headings to define topics and subtopics of information. You can define four levels of headings:

H1 Centers text in the information region. Use this heading level to identify a main topic of information.

H2, H3, H4
Formats text against the left margin of the information region. Use one of these heading levels to identify subtopics of information.

You must code headings sequentially. The conversion utility adds a blank line to the information region before and after the formatted heading text. The heading tags have no attributes associated with them, and they don't require an end tag.

The following markup contains an information region using two heading levels and paragraphs following each one.

```
<!doctype dm system>
<panel name=infopan3 width=42>Information
  <area>
    <info width=40>
      <h1>A Main Topic
      <p>Notice how the heading is in the
      center of the information region?
        <h2>A Subtopic
        <p>This heading is left-justified.
        <h2>Another Subtopic
        <p>Here's another level-two heading.
    </info>
  </area>
</panel>
```

Here's the formatted result:

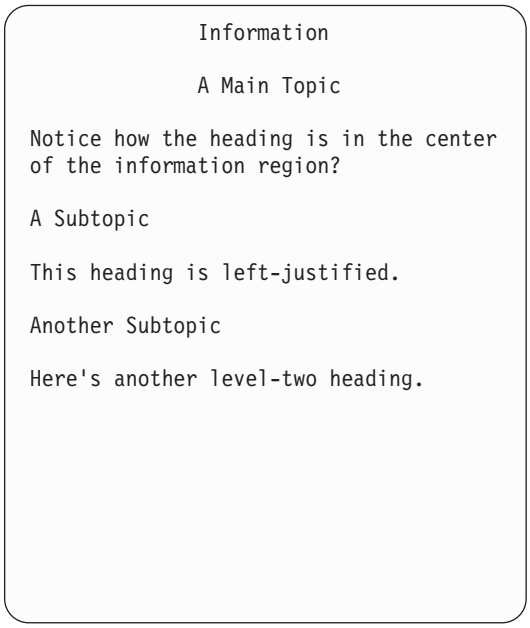


Figure 46. Headings (H1-H2)

Lines

Occasionally, you'll want to present text that you don't want formatted by the compiler, or that you want to show "as is". You can use the LINES (lines) tag and its required end tag to do this. All text coded within a LINES definition is treated as unformatted text, and you can position the text however you like on each line. If the text line is too long to fit in the available width, the conversion utility truncates the text and issues a warning message.

The LINES tag requires an end tag.

There are many ways to use a LINES definition. Here we use it for a quotation:

```
<!doctype dm system>
<panel name=specact width=48>Special Activities
  <area>
    <info width=46>
      <lines>
      Between the dark and daylight,
      When the night is beginning to lower,
      Comes a pause in the days' occupations,
      That's known as the children's hour.
                              -Longfellow
      </lines>
      <p>Every Tuesday evening at seven
      o'clock, we present the Children's Hour,
      a one-hour recital of selected children's
      stories in our children's section.
    </info>
  </area>
</panel>
```

And our quotation appears just the way we marked it up:

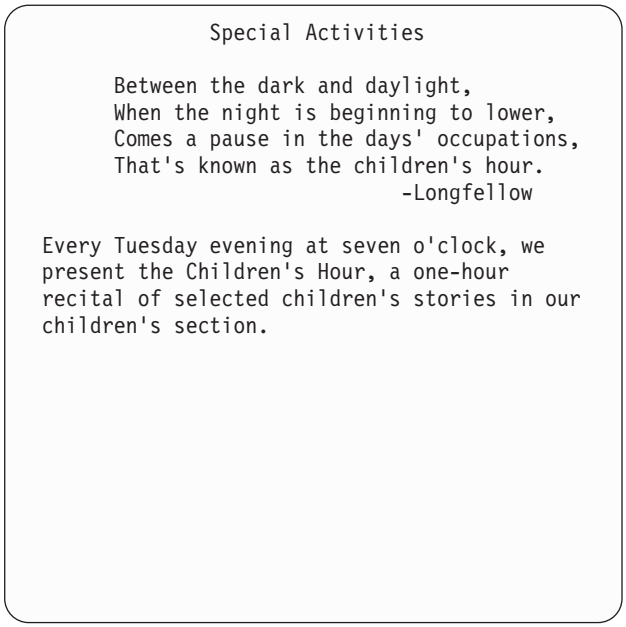


Figure 47. LINES

Examples

The XMP (example) tag is similar to the LINES tag, in that it allows you to code unformatted text. However, the text of an XMP definition is indented two spaces from the current margin, as opposed to the text of a LINES definition, which is not indented from the current margin.

Like a LINES definition, you should avoid coding lines of text in an XMP definition that exceed the available formatting width of the information region. If the text exceeds the defined width, it is truncated.

The XMP tag requires a matching end tag.

Here's the formatted result of an example using the XMP tag:

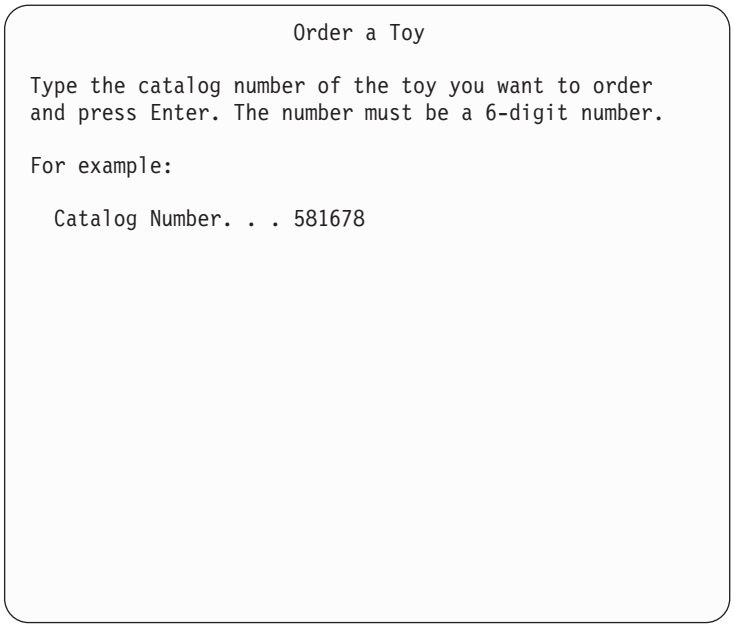


Figure 48. XMP

The markup for the previous panel looks like this:

```
<!doctype dm system>
<panel name=toy1 width=57>Order a Toy
<area>
<info width=55>
<p>Type the catalog number of the toy you want to order
and press Enter.
The number must be a 6-digit number.
<p>For example:
<xmp>
Catalog Number. . . 581678
</xmp>
</info>
</area>
</panel>
```

Figures

The FIG (figure) tag is yet another way you can code text that isn't formatted. It works just like the LINES tag, except you can add a ruled border above and below the figure to separate it from the rest of the panel. You can also provide a caption for the figure using the FIGCAP tag.

Like the LINES and XMP tags, the FIG tag requires an end tag.

To define the ruled borders for the figure, use the FRAME attribute of the FIG start tag. The FRAME attribute has two values, RULE, which is the default, and NONE. Because RULE is the default value, you don't need to specify this attribute if you want ruled lines above and below the figure. To create a figure without rules, specify NONE as the FRAME value.

The figure in this panel formats with a ruled border:

```
<!doctype dm system>
<panel name=toy2 width=57>Order a Toy
<area>
```

```
<info width=55>
<p>Type the catalog number of
the toy you want to order and
press Enter.
The number must be a 6-digit number.
<p>For example:
<xmp>
Catalog Number. . . 581678
</xmp>
<p>A description of the toy will appear.
<fig>
          ZOOM-A-GO DAREDEVIL SET

  Your kids will have hours of excitement
  playing with this full set of action toys.
  Requires 80 "AA" batteries.  Not included.
</fig>
</info>
</area>
</panel>
```

Here's the formatted panel:

```
                      Order a Toy

Type the catalog number of the toy you want to order
and press Enter. The number must be a 6-digit number.

For example:

  Catalog Number. . . 581678

A description of the toy will appear.

-----------------------------------------------

          ZOOM-A-GO DAREDEVIL SET

  Your kids will have hours of excitement
  playing with this full set of action toys.
  Requires 80 "AA" batteries.  Not included.

-----------------------------------------------
```

Figure 49. Figure with Rules

If we wanted the figure to appear without a ruled border, we would have specified FRAME=NONE for the FIG tag.

The FIG tag also has an optional WIDTH attribute that allows you to specify how the figure is aligned in the information region. The valid values for WIDTH are PAGE and COL. PAGE, which is the default value, aligns the figure along the left margin of the information region. COL indicates that the figure is aligned along the current left margin; that is, the current margin defined by the tag the figure is nested in. This is useful, for example, for aligning figures within list items.

Figure Captions (FIGCAP) Tag

To add a caption to the figure in Figure 49, use a FIGCAP tag and caption text within the figure definition, like this:

```
<!doctype dm system>
<panel name=toy3 width=57>Order a Toy
<area>
<info width=55>
<p>Type the catalog number of
the toy you want to order and
press Enter.
The number must be a 6-digit number.
<p>For example:
<xmp>
Catalog Number. . . 581678
</xmp>
<p>A description of the toy will appear.
<fig>
          ZOOM-A-GO DAREDEVIL SET

  Your kids will have hours of excitement
  playing with this full set of action toys.
  Requires 80 "AA" batteries.  Not included.
<figcap>Zoom-A-Go Daredevil Set
</fig>
</info>
</area>
</panel>
```

The figure caption appears just below the bottom figure rule:

```
                      Order a Toy

Type the catalog number of the toy you want to order
and press Enter. The number must be a 6-digit number.

For example:

  Catalog Number. . . 581678

A description of the toy will appear.

-----------------------------------------------

          ZOOM-A-GO DAREDEVIL SET

  Your kids will have hours of excitement
  playing with this full set of action toys.
  Requires 80 "AA" batteries.  Not included.

-----------------------------------------------
Zoom-A-Go Daredevil Set
```

Figure 50. Figure Caption

Defining Lists

Sometimes you want to present information to the user that is not appropriate in paragraph form, such as list items, a sequence of items or actions, or definitions. For these situations (and many others), you can use the DTL list tags to format your text appropriately.

You can create these types of lists:

Note lists	Format as numbered lists of notes under a header called **Notes**.
Simple lists	Format as indented lists of items without any preceding identifiers.
Unordered lists	Format as indented lists of items with each item preceded by a bullet (o), a hyphen (-), or dashes (--), depending on the level of nesting.
Ordered lists	Format as indented lists of items with each item preceded by a number or letter indicating its sequence in the list.
Definition lists	Format in two columns, with terms in one column and their matching descriptions in the other. You can also specify headings for each column in the list. (This list is a definition list.)
Parameter lists	Format in two columns. This list is specifically designed to identify and define parameter terms.

The list items in note lists, simple lists, unordered lists, and ordered lists are created with the list item (LI) tag. The LI tag does not require an end tag. It is implicitly ended by another LI tag, an LP tag, or the end tag of the list it is coded within.

Note Lists

See "Alerting Users: Notes, Warnings, Cautions, and Attention" on page 139 for an example showing the use of note lists.

Simple Lists

A simple list is the least complex type of list. Use a simple list when the information you are presenting does not follow a sequential pattern or when bullets are not required to discriminate one list item from another.

Figure 51 on page 125 illustrates a simple list.

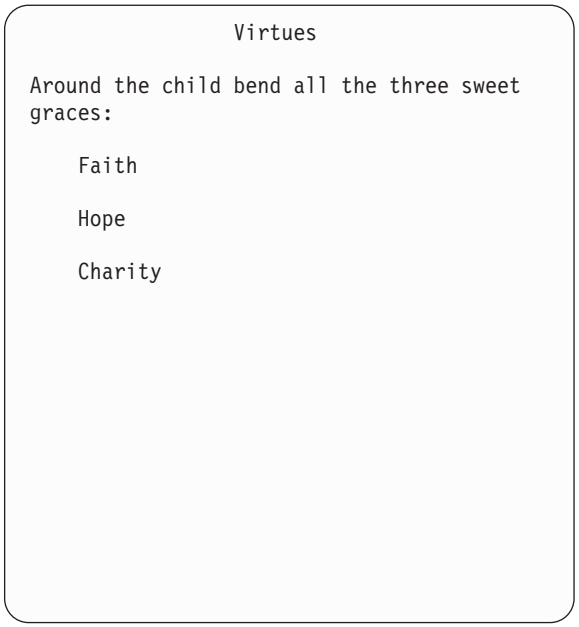


Figure 51. Simple List

This is the markup for the panel:

```
<!doctype dm system>
<panel name=slistx1 width=44>Virtues
  <area>
    <info width=42>
      <p>Around the child bend all the
      three sweet graces:
      <sl>
        <li>Faith
        <li>Hope
        <li>Charity
      </sl>
    </info>
  </area>
</panel>
```

We used the SL tag and its matching end tag to define the simple list. We defined each of the list items by nesting the LI tags within the simple list definition.

As you can see, our simple list formatted with a blank line between each of the list items. For cases where you need to conserve space, you can use the COMPACT attribute to format the list without blank lines between the list items.

Code the COMPACT attribute within the SL start tag (before the tag close delimiter), like this:

```
<!doctype dm system>
<panel name=slistx2 width=44>Virtues
  <area>
    <info width=42>
      <p>Around the child bend all the
      three sweet graces:
      <sl compact>
        <li>Faith
        <li>Hope
        <li>Charity
```

```
      </sl>
    </info>
  </area>
</panel>
```

Now the simple list is compacted:

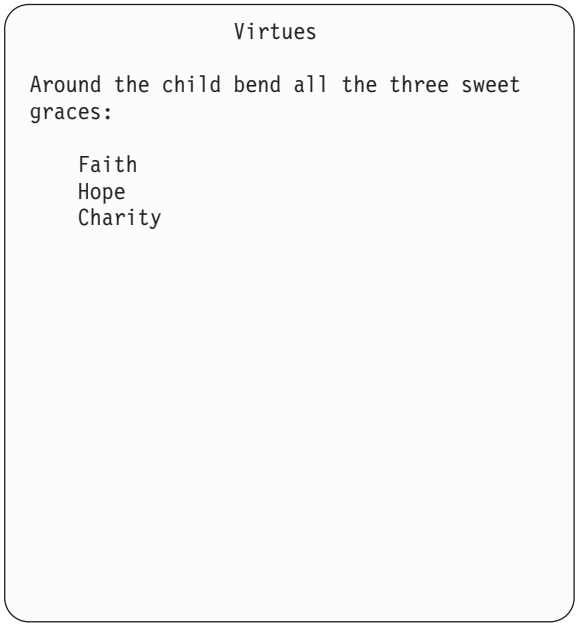


Figure 52. Compact Simple List

You can also nest simple lists within other lists. The list items format at different indentation levels, based on the level of nesting.

The indentation for the list item is based on the SPACE attribute of the LI tag and the enclosing list tag. When SPACE=NO (or the SPACE attribute is not present) the list item indentation is 4 spaces. When SPACE=YES, the indentation is 3 spaces. See "Chapter 13. Tag Reference" on page 211 for additional information about the LI, SL, OL, and UL tags.

Unordered Lists

Unordered lists are similar to simple lists, except each list item is preceded by symbol that is dependent on the nesting level of the list. You don't have to supply the symbols–the conversion utility does that for you.

Use an unordered list if the list items are long and you don't want to imply any particular sequence in the list.

Here's an unordered list:

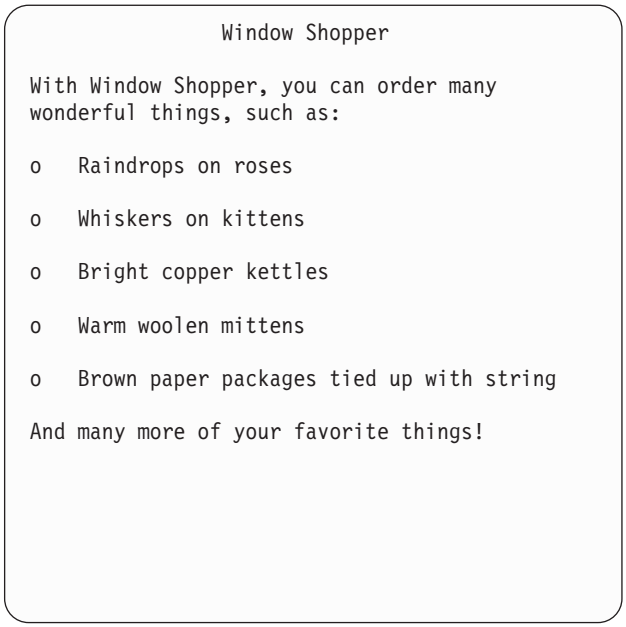


Figure 53. Unordered List

And here's the markup for this unordered list:

```
<!doctype dm system>
<panel name=winshop width=48>Window Shopper
  <area>
    <info width=46>
      <p>With Window Shopper, you can order many wonderful things,
      such as:
      <ul>
        <li>Raindrops on roses
        <li>Whiskers on kittens
        <li>Bright copper kettles
        <li>Warm woolen mittens
        <li>Brown paper packages tied up with string
      </ul>
      <p>And many more of your favorite things!
    </info>
  </area>
</panel>
```

For our unordered list, we used the UL tag and its matching end tag. As you can see, even though we didn't code the bullet symbols (o) in the markup, they appear before each of the list items in the unordered list.

We could make this list compact like our simple list example because the COMPACT attribute is also valid for the UL tag. Likewise, we could use the SPACE attribute to control indentation of the list items for the UL tag.

You can also define levels of unordered lists; that is, you can nest unordered lists within other unordered lists. When you do this, the symbols preceding the list items in each level of the list vary, depending on the level of nesting. Specifically, the list items in the first (or only) level of unordered list are preceded by bullets (o), as shown in Figure 53. If you nest another unordered list within an unordered list,the list items in that list are preceded by hyphen symbols (-). A third-level unordered list has dashes (--) preceding the list items. The nested tag text is aligned according to the level of nesting.

To show how this works, we'll create a panel with three levels of unordered lists.

```
<!doctype dm system>
<panel name=ulists width=42>Nested Unordered Lists
  <area>
    <info width=40>
      <ul>
        <li>First level, first item
        <li>First level, second item
         <ul>
           <li>Second level, first item
           <li>Second level, second item
             <ul>
               <li>Third level, only item
             </ul>
         </ul>
        <li>Back to the first level
      </ul>
    </info>
  </area>
</panel>
```

Here's how this panel looks:

```
          Nested Unordered Lists

o   First level, first item

o   First level, second item

    -   Second level, first item

    -   Second level, second item

        --  Third level, only item

o   Back to the first level
```

Figure 54. Nested Unordered Lists

If you nest more than three levels of unordered lists, the sequence of bullets, hyphens, and dashes repeats. For example, a fourth level would be preceded by bullets, a fifth level by hyphens, and so on.

Remember, all lists must be explicitly ended with the appropriate list end tag.

Ordered Lists

Ordered lists imply an outline sequence to the list items by preceding each of the list items with a number or character depending on the level of nesting.

Here's an ordered list:

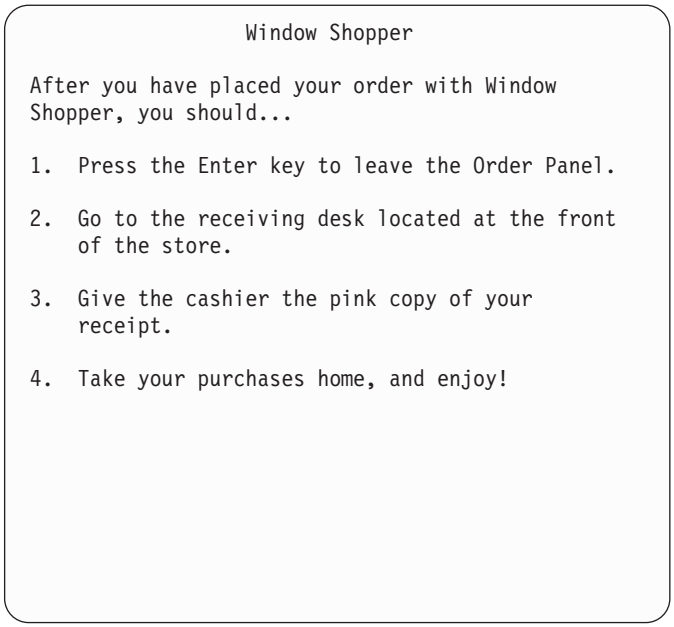


Figure 55. Ordered List

You don't supply the numbers for the list items in your markup; they are generated automatically. This saves you time when you revise ordered lists, because you can insert, delete, or rearrange list items without renumbering them yourself.

Here's the markup we used for this list:

```
<!doctype dm system>
<panel name=winshop2 width=52>Window Shopper
  <area>
    <info width=50>
      <p>After you have placed your order with Window Shopper, you should...
      <ol>
        <li>Press the Enter key to leave the Order Panel.
        <li>Go to the receiving desk located at the front of the store.
        <li>Give the cashier the pink copy of your receipt.
        <li>Take your purchases home, and enjoy!
      </ol>
    </info>
  </area>
</panel>
```

Like other types of lists, you can nest ordered lists within other lists. And, like unordered lists, the levels of the lists you nest determine the characters that precede the list items.

Specifically, the conversion utility uses the following sequence when processing list items in nested ordered lists:

- First-level list items are preceded by sequential numbers followed by a period and 2 spaces [5].

5. The default indentation for a list item is 4 spaces. When the SPACE=YES attribute is coded, the indentation is 3 spaces. See the LI and OL tag descriptions in "Chapter 13. Tag Reference" on page 211 for more information.

- Second-level list items are preceded by sequential lowercase alphabetic characters followed by a period and 2 spaces [5].
- Third-level list items are preceded by sequential numbers followed by a close parentheses symbol and 2 spaces [5].
- Fourth-level list items are preceded by sequential lowercase alphabetic characters followed by a close parentheses symbol and 2 spaces [5].

Note: Each level beyond the first level indents 4[5] spaces.

The sequence of nesting is repeated for levels of nesting beyond the fourth level. For example, the list items in a fifth level of nesting are preceded by sequential numbers followed by a period.

To show you what this looks like, we'll nest three levels of ordered lists in this markup. We'll use the COMPACT attribute in the third level to conserve space.

```
<!doctype dm system>
<panel name=olists width=42>Nested Ordered Lists
  <area>
    <info width=40>
      <ol>
        <li>Step one (first level)
        <li>Step two (first level)
          <ol>
            <li>Step one (second level)
            <li>Step two (second level)
              <ol compact>
                <li>Step one (third level)
                <li>Step two (third level)
              </ol>
            <li>Step three (second level)
          </ol>
        <li>Step three (first level)
      </ol>
    </info>
  </area>
</panel>
```

Here's how the DTL compiler formats this panel:

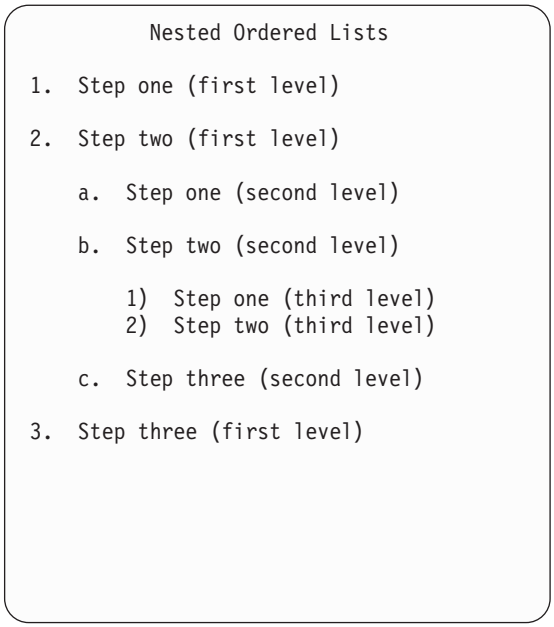


Figure 56. Nested Ordered Lists

Definition Lists

Definition lists allow you to identify a list of words or phrases and their corresponding definitions. A simple definition list formats as a two-column list: the terms you define appear in the left column, and the definitions for the terms appear in the right column. Definition lists are slightly more complex than the previous lists we've discussed, because of the additional tags required to construct them.

The tags used to create a definition list are:

DL Begins a definition list. The required end tag ends the list.

DT Identifies the term being defined. The definition term is formatted in the left column of the list. It does not require an end tag.

DD Identifies the term description. Each definition description is formatted in the right column of the list, immediately opposite or below its associated term. It does not require an end tag.

You can also create headings for definition list columns. There are two additional tags that you can use to do this. They are:

DTHD Defines a header for the definition term column.

DDHD Defines a header for the definition description column.

Both of these tags are optional for creating definition lists. We'll show you how you can use them to enhance definition lists later on in this section.

Here's an example of a definition list:

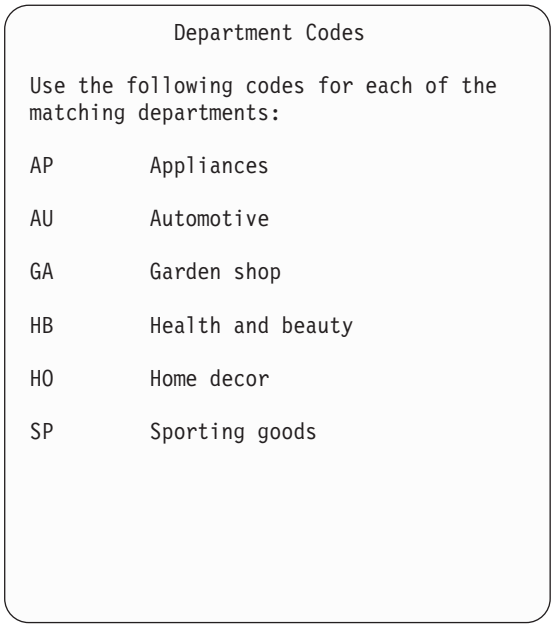


Figure 57. Definition List

Here's the markup:

```
<!doctype dm system>
<panel name=deptcode width=42>Department Codes
  <area>
    <info width=40>
      <p>Use the following codes for each of the
      matching departments:
      <dl>
        <dt>AP
        <dd>Appliances
        <dt>AU
        <dd>Automotive
        <dt>GA
        <dd>Garden shop
        <dt>HB
        <dd>Health and beauty
        <dt>HO
        <dd>Home decor
        <dt>SP
        <dd>Sporting goods
      </dl>
    </info>
  </area>
</panel>
```

A definition list can contain multiple definition terms. The TSIZE attribute of the enclosing DL tag specifies the number of DT tags in a group and their respective widths. For example, TSIZE='10 5' specifies 2 definition term columns with sizes of 10 and 5 characters, respectively.

The DL tag has optional attributes:

TSIZE specifies the space allocated for the term column or columns

BREAK indicates if the definition formats on the same line as its associated term

COMPACT determines if there is a space between each set of terms and descriptions

NOSKIP suppresses the blank line normally placed before the list

INDENT controls the indentation from the current left margin

FORMAT controls the location of the definition term within the TSIZE space

Use the TSIZE attribute to specify how much space you want for the definition term column (or columns). The default value is 10 bytes, which also sets the default number of DT tags to **one**. If you want to specify more (or less) space than the default, or multiple DT tags, use the TSIZE attribute to assign the value you want.

Use the BREAK attribute to specify where the definition descriptions are supposed to start (on the same line as the definition terms or on the next line). The BREAK attribute can be specified as NONE, ALL, or FIT.

NONE The definition descriptions start on the same lines as the definition terms.

ALL All of the definition descriptions start on the line after the definition terms.

FIT The definition descriptions are to start on the next line only when the definition term does not fit in the allocated space and spills over into the description area.

The definition list in Figure 57 on page 132 used the default BREAK=NONE. We'll define another list that uses BREAK=ALL.

```
<!doctype dm system>
<panel name=reverb1 width=52>Reverberations
  <area>
    <info width=50>
      <p>Reverberations is one of the most popular brands of electronic
      components available today.
      We stock the following Reverberations components:
      <dl break=all>
        <dt>CD Player Unit
        <dd>With auto-search, auto-off, power door, and
        a two-year warranty.
        <dt>Receiver
        <dd>Digital, 6-speaker hookup, and built-in
equalizer.
        <dt>Tape deck
        <dd>Supports metal and chrome cassettes, and comes with
        a two-year warranty.
      </dl>
    </info>
  </area>
</panel>
```

Figure 58 on page 134 shows how this definition list formats.

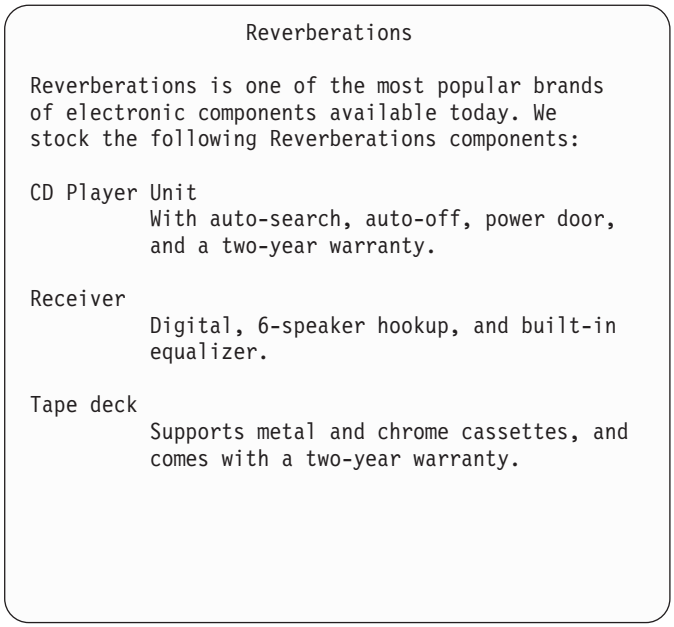


Figure 58. Definition List (BREAK=ALL)

Because the TSIZE and BREAK attributes lend versatility to definition lists, we can rearrange this list practically any way we want. We'll change the BREAK value to FIT, and increase the TSIZE to 13 to show you what we mean. We'll also add headings to the list to show you how they format.

```
<!doctype dm system>
<panel name=reverb2 width=52>Reverberations
  <area>
    <info width=50>
      <p>Reverberations is one of the most popular brands of electronic
      components available today.
      We stock the following Reverberations components:
      <dl tsize=13 break=fit>
        <dthd>Component
        <ddhd>Features
        <dt>CD Player Unit
        <dd>With auto-search, auto-off, power door, and
        a two-year warranty.
        <dt>Receiver
        <dd>Digital, 6-speaker hookup, and built-in equalizer.
        <dt>Tape deck
        <dd>Supports metal and chrome cassettes, and comes with
        a two-year warranty.
      </dl>
    </info>
  </area>
</panel>
```

Here's how the panel looks now:

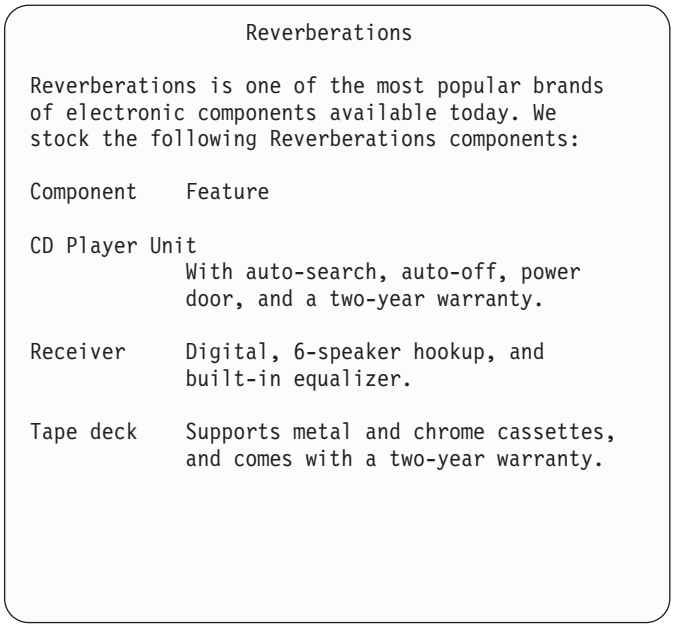


Figure 59. Definition List (BREAK=FIT)

Parameter Lists

Parameter lists are another way of defining terms in a list form. You use a parameter list when the terms you are defining are related to the application in some way (for example, showing codes or parameters).

The tags you use to create parameter lists are the PARML tag and its required end tag, the PT (parameter term) tag, and the PD (parameter description) tag. The parameter list tags work a lot like the definition list tags in defining terms and descriptions, except there are no tags for defining list headings.

The PARML tag also contains the TSIZE, BREAK, COMPACT, INDENT, and SKIP attributes. The TSIZE default value is 10 bytes, as it is for definition lists. However, the BREAK default value for parameter lists is ALL, instead of NONE, as in definition lists. Thus, the parameter descriptions format on the lines following the parameter terms unless you specify otherwise.

A parameter list can contain multiple parameter terms. The TSIZE attribute of the enclosing PARML tag specifies the number of PT tags in a group and their respective widths. For example, TSIZE='10 5' specifies 2 parameter term columns with sizes of 10 and 5 characters, respectively.

Here's the markup for a typical parameter list:

```
<!doctype dm system>
<panel name=ordnum width=52>Order Numbers
  <area>
    <info width=50>
      <p>The order number assigned to each inventory item
      represents specific information about the item.
      <p>Specifically,
      <parml>
        <pt>123
```

```
        <pd>The first 3 digits represent the
        department the item is stocked in.
        <pt>456
        <pd>The fourth, fifth, and sixth digits
        represent the item.
        <pt>78
        <pd>The seventh and eighth digits represent
        the lot number of the item.
      </parml>
    </info>
  </area>
</panel>
```

And here's the formatted parameter list:

```
                  Order Numbers

The order number assigned to each inventory item
represents specific information about the item.

Specifically,

123
          The first 3 digits represent the
          department the item is stocked in.

456
          The fourth, fifth, and sixth digits
          represent the item.

78
          The seventh and eighth digits represent
          the lot number of the item.
```

Figure 60. Parameter List

Nesting Tags within Lists

The format of your lists isn't confined to only list items. You can also nest other tags within the list items. For example, if a list item requires an additional paragraph, you can nest a P tag following the list item.

This markup contains an ordered list with a paragraph nested within the second list item.

```
<!doctype dm system>
<panel name=winshop3 width=52>Window Shopper
  <area>
    <info width=50>
      <p>After you have placed your order with Window Shopper, you should...
      <ol>
        <li>Press the Enter key to leave the Order Panel.
        <li>Go to the receiving desk located at the front of the store.
          <p>Don't forget to bring your receipt!
        <li>Give the cashier the pink copy of your receipt.
        <li>Take your purchases home, and enjoy!
```

```
      </ol>
    </info>
  </area>
</panel>
```

The paragraph text follows the indentation of the preceding list item, like this:

Figure 61. Nested Paragraph within a List

The List Part (LP) Tag

If you want to insert unindented text in a list, use the list part (LP) tag. The LP tag is useful for providing information about the list items that follow it.

We added a list part to the panel shown in Figure 61:

```
<!doctype dm system>

<panel name=winshop4 width=52>Window Shopper
  <area>
    <info width=50>
      <p>After you have placed your order with Window Shopper, you should...
      <ol>
        <li>Press the Enter key to leave the Order Panel.
        <li>Go to the receiving desk located at the front of the store.
          <p>Don't forget to bring your receipt!
        <li>Give the cashier the pink copy of your receipt.
          <lp>Occasionally, the item you ordered won't be in stock.
          If this occurs, the cashier will be happy to delete
          the item from your order.
        <li>Take your purchases home, and enjoy!
      </ol>
    </info>
  </area>
</panel>
```

Here's the formatted result:

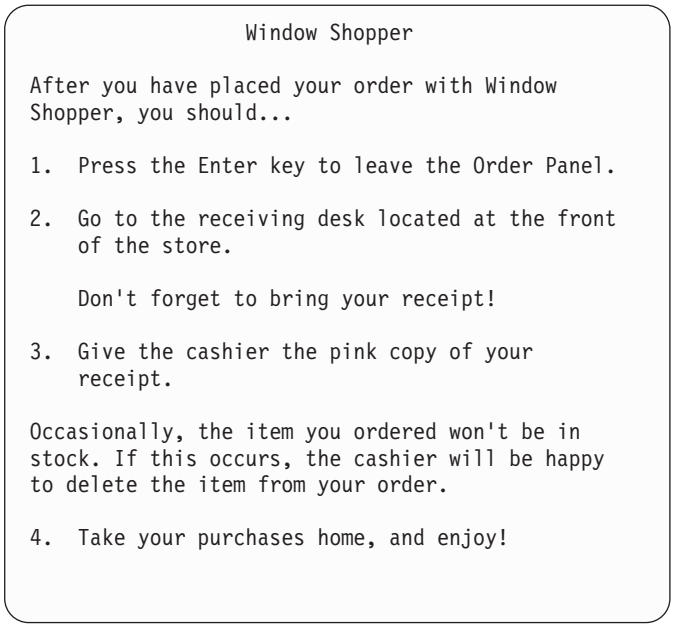


Figure 62. List Part

Nesting Lists within Lists

On pages 127 and 129 we showed you how to define levels of nested unordered and ordered lists. You can also nest different types of lists within other lists.

Here's an example of an unordered list nested within a definition list:

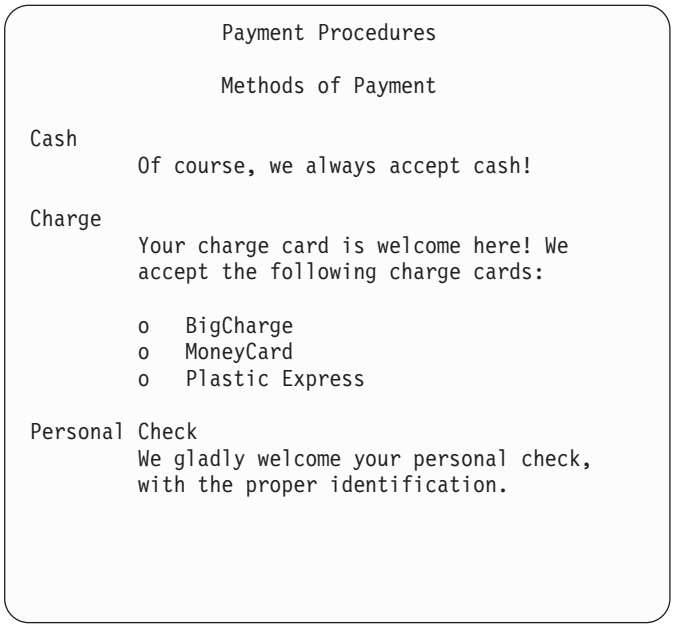


Figure 63. Nested Unordered List in a Definition List

Here's the markup we used to create the nested lists in Figure 63 on page 138:

```
<!doctype dm system>

<panel name=payment width=52>Payment Procedures
  <area>
    <info width=50>
      <h1>Methods of Payment
      <dl tsize=9 break=all>
        <dt>Cash
        <dd>Of course, we always accept cash!
        <dt>Charge
        <dd>Your charge card is welcome here!
        We accept the following charge cards:
          <ul compact>
            <li>BigCharge
            <li>MoneyCard
            <li>Plastic Express
          </ul>
        <dt>Personal check
        <dd>We gladly welcome your personal check,
        with the proper identification.
      </dl>
    </info>
  </area>
</panel>
```

You can nest any type of list within another list. Remember, whenever you nest lists, be sure that you end each level with its proper end tag.

Alerting Users: Notes, Warnings, Cautions, and Attention

DTL provides you with tags that you can use to alert the user to certain text that warrants special attention. Whether you are noting a minor aspect of the application or alerting the user to the risk of possible damage to programs or data, you can alert the user appropriately.

In this section, we discuss the following tags:

- ATTENTION
- CAUTION
- NOTE
- NOTEL
- NT
- WARNING.

Notes

The NOTE, NOTEL, and NT tags format as noted text. Use notes to emphasize minor points.

When you use either the NOTE or NT tag, you get the text "Note:" followed by a space before the text you specify. However, the text is formatted differently depending on which tag you use.

The NOTEL tag is formatted with the first line containing the text ″Notes:″ followed by a numbered list of note information provided by the <LI> tag.

The NOTE tag

If the text is a single paragraph, you use the NOTE tag. The text is formatted as an unindented block, like a paragraph. The NOTE tag does not require a matching end tag.

You use the NOTE tag like this:

```
<!doctype dm system>
<panel name=widget61 width=50>Widgets
  <area>
    <info width=48>
      <p>Choose the type of Widget you want to order by placing
      the cursor on the field and pressing Enter.
      <note>If the Widget you wish to order is not in stock, please
      refer to the "Back Order" panel to place an order.
    </info>
  </area>
</panel>
```

Figure 64 shows how it formats.

```
                    Widgets

Choose the type of Widget you want to order by
placing the cursor on the field and pressing
Enter.

Note: If the Widget you wish to order is not in
stock, please refer to the "Back Order" panel to
place an order.
```

Figure 64. Note (NOTE tag)

The NOTEL tag

If more than one note is used for special attention information, you use the NOTEL tag. Each note is provided by a separate LI tag. The notes are numbered similar to the format described in "Ordered Lists" on page 128. You use either the P or LP tag to add any additional paragraphs in the NOTEL definition. Use the required end tag to end the NOTEL definition.

In this example, 2 notes are used, 1 with more than one paragraph. We use the NOTEL tag and its required end tag along with LI tags to define the notes, and a P tag for the additional paragraph.

```
<!doctype dm system>
<panel name=widget63 width=50>Widgets
  <area>
    <info width=48>
      <p>Choose the type of Widget you want to order by placing
      the cursor on the field and pressing Enter.
      <notel>
        <li>If the Widget you wish to order is not in stock, please
            refer to the "Back Order" panel to place an order.
        <li>Back-ordered Widgets usually arrive within three days.
          <p>Please check again in three days.
```

```
      </notel>
      <p>If you want to order more than one Widget, specify the quantity
      and press Enter.
    </info>
  </area>
</panel>
```

Notice that the P tag in the note is coded *before* the NOTEL end tag, indicating that the second paragraph belongs in the note.

Here's how the panel looks now:

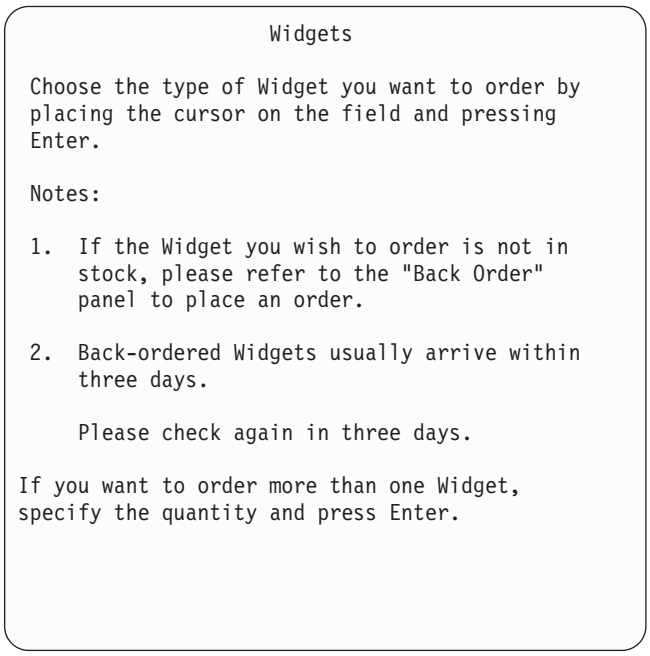


Figure 65. Notel (NOTEL tag)

As you can see, the the text of the NOTEL tag is formated as a list under the ″Notes:″ heading. The text of the P tag is indented to match the list items.

The NT tag

If the note requires more than one paragraph, you use the NT tag. You use the P tag to add any additional paragraphs in the NT definition. Use the required end tag to end the NT definition.

Another difference between the NOTE and NT tag is that the NT tag indents the note text from the left panel margin.

In this example, the note is longer than one paragraph. We use the NT tag and its required end tag to define the note, and a P tag for each additional paragraph.

```
<!doctype dm system>
<panel name=widget62 width=50>Widgets
  <area>
    <info width=48>
      <p>Choose the type of Widget you want to order by placing
      the cursor on the field and pressing Enter.
      <nt>If the Widget you wish to order is not in stock, please
      refer to the "Back Order" panel to place an order.
```

```
        <p>Back-ordered Widgets usually arrive within three days.
      </nt>
      <p>If you want to order more than one Widget, specify the quantity
      and press Enter.
    </info>
  </area>
</panel>
```

Notice that the P tag in the note is coded *before* the NT end tag, indicating that the second paragraph belongs in the note.

Here's how the panel looks now:

```
                    Widgets

Choose the type of Widget you want to order by
placing the cursor on the field and pressing
Enter.

Note: If the Widget you wish to order is not in
      stock, please refer to the "Back Order"
      panel to place an order.

      Back-ordered Widgets usually arrive
      within three days.

If you want to order more than one Widget,
specify the quantity and press Enter.
```

Figure 66. Note (NT tag)

As you can see, the text of the NT tag is indented, as is the text of the P tag coded within the NT tag.

Attention and Warning

Attention statements and warning statements alert the user of a possible risk involved with a user action, or of existing error conditions.

You must immediately precede the ATTENTION or WARNING tag with a P (paragraph) tag, LI (list item) tag, or LP (list part) tag. The warning statement formats with the term "Warning:" before the text. The attention statement formats with the term "Attention:" before the text.

The ATTENTION and WARNING tags have no associated attributes and require a matching end tag.

Here's the markup for a warning statement that formats as a paragraph.

```
<!doctype dm system>
<panel name=addfile width=50>Changing a File
  <area>
    <info width=48>
```

```
      <p>After you have made the desired changes
      to the file, press Enter to save the changes.
      <p><warning>Pressing Enter saves
      ALL changes made to the file.
      You can cancel this operation by pressing
      the F12=Cancel key.
      </warning>
    </info>
  </area>
</panel>
```

Here's the formatted result:

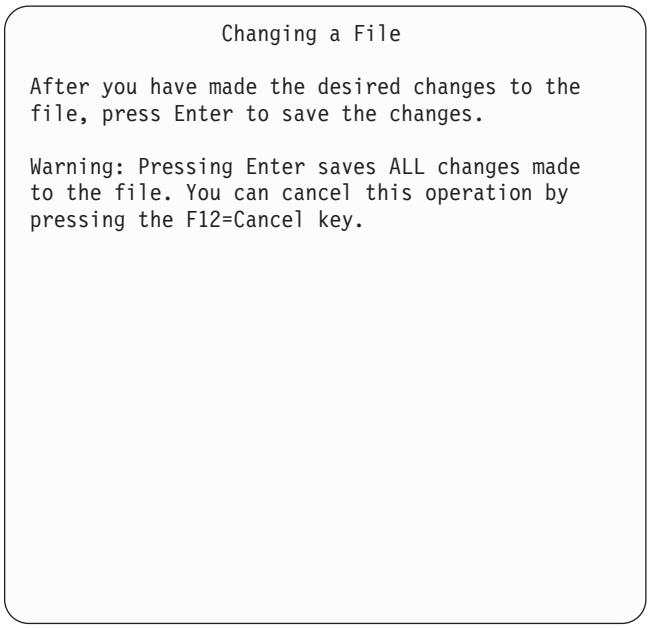


Figure 67. Warning

Caution

Caution statements indicate the greatest degree of severity. Like the WARNING tag, the CAUTION tag has a required end tag, and must be preceded by a P (paragraph) tag, LI (list item) tag, or LP (list part) tag. The caution statement formats with the term "CAUTION:" followed by the caution text on the next line.

```
<!doctype dm system>
<panel name=delfile width=50>Deleting a File
  <area>
    <info width=48>
      <p>To delete a file, type the filename in the
      "Delete this file" field and press Enter.
      <p>A message appears asking for verification.
      To continue the delete operation, press Enter.
      <p><caution>Verifying the delete operation
      permanently deletes the file from your records.
      There is no chance of recovery.
      </caution>
    </info>
  </area>
</panel>
```

Here's the formatted result:

Figure 68. Caution

Emphasizing Panel Text

You can emphasize text on application panels or on help panels with highlighting by using the HP (highlighted phrase) tag. You can also highlight words or phrases to indicate that additional information is available by using the RP (reference phrase) tags. On a color terminal, the emphasized text displays in a CUA defined color, or whatever color you set with the Color Change Utility.

Highlighting requires the use of 3270 attribute bytes to control the display of highlighted text. The attribute positions before and after the highlighted text display as blank spaces. These attributes might limit the formatting of your highlighted phrase or reference phrase.

For example, if you typed your text as follows:

```
<HP>To cancel this option</HP>, press the F12 key.
```

Here's the result:

To cancel this option, press the F12 key.

You can prevent this situation by writing statements that do not require punctuation following an HP or an RP end tag.

Highlighted Phrases

The HP (highlighted phrase) tag provides emphasis through highlighting. You can focus the user's attention to particular sections of the panel text by highlighting words, phrases, or entire paragraphs.

The HP tag requires a matching end tag to indicate the end of a highlighted phrase.

The following markup example shows you how to use the HP tag.

```
<!DOCTYPE DM SYSTEM>

 <VARCLASS NAME=date    TYPE='char 8'>
 <VARCLASS NAME=numcls  TYPE='numeric 7'>
 <VARCLASS NAME=namecls TYPE='char 25'>
 <VARCLASS NAME=char1cls TYPE='char 1'>
 <VARCLASS NAME=char7cls TYPE='char 7'>

 <VARLIST>
   <VARDCL NAME=whchsrch VARCLASS=char1cls>
   <VARDCL NAME=curdate VARCLASS=date>
   <VARDCL NAME=cardno  VARCLASS=numcls>
   <VARDCL NAME=name    VARCLASS=namecls>
   <VARDCL NAME=address VARCLASS=namecls>
   <VARDCL NAME=cardsel VARCLASS=char1cls>
   <VARDCL NAME=card  VARCLASS=char7cls>
   <VARDCL NAME=north VARCLASS=char1cls>
   <VARDCL NAME=south VARCLASS=char1cls>
   <VARDCL NAME=east  VARCLASS=char1cls>
   <VARDCL NAME=west  VARCLASS=char1cls>
 </VARLIST>

<PANEL NAME=hlitep>Library Card Registration
<AB>
 <ABC>File
   <PDC>Add Entry
     <ACTION RUN=add>
   <PDC>Delete Entry
     <ACTION RUN=delete>
   <PDC>Update Entry
     <ACTION RUN=update>
   <PDC>Exit
     <ACTION RUN=exit>
 <ABC>Search
   <PDC CHECKVAR=whchsrch MATCH=1>Search on name
     <ACTION SETVAR=whchsrch VALUE=1>
     <ACTION RUN=search>
   <PDC CHECKVAR=whchsrch MATCH=2>Search on card number
     <ACTION SETVAR=whchsrch VALUE=2>
     <ACTION RUN=search>
  <ABC>Help
    <PDC>Extended Help...
      <ACTION RUN=exhelp>
    <PDC>Keys Help...
      <ACTION RUN=keyshelp>
 </AB>
<TOPINST>Type in <HP>patron's name</HP> and <HP>card number</HP>
         (if applicable).
<TOPINST>Then select an action bar choice.
<AREA>
    <DTACOL PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25 SELWIDTH=25>
    <DTAFLD DATAVAR=curdate USAGE=out ENTWIDTH=8>Date
    <DTAFLD DATAVAR=cardno ENTWIDTH=7>Card No.
      <DTAFLDD>(A 7-digit number)
    <DTAFLD DATAVAR=name>Name
      <DTAFLDD>(Last, First, M.I.)
    <DTAFLD DATAVAR=address>Address
     </DTACOL>
  <DIVIDER>
  <REGION DIR=horiz>
  <SELFLD NAME=cardsel PMTWIDTH=30 SELWIDTH=38>Choose
  one of the following
    <CHOICE CHECKVAR=card MATCH=new>New
    <CHOICE CHECKVAR=card MATCH=renew>Renewal
    <CHOICE CHECKVAR=card MATCH=replace>Replacement
  </SELFLD>
  <SELFLD TYPE=multi PMTWIDTH=30 SELWIDTH=25>Check valid branches
```

```
    <CHOICE NAME=north HELP=nthhlp CHECKVAR=nth>North Branch
    <CHOICE NAME=south HELP=sthhlp CHECKVAR=sth>South Branch
    <CHOICE NAME=east HELP=esthlp CHECKVAR=est>East Branch
    <CHOICE NAME=west HELP=wsthlp CHECKVAR=wst>West Branch
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

Here's the formatted result:

```
  File  Search  Help
------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number (if applicable).

Then select an action bar choice.

Date . . . : 08/29/90
Card No. . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following          Check valid branches
__  1.  New                         _  North Branch
    2.  Renewal                     _  South Branch
    3.  Replacement                 _  East Branch
                                    _  West Branch

Enter a command ===> _____________________________________________________
 F1=Help        F2=Split       F3=Exit        F6=KEYSHELP    F9=Swap
F12=Cancel
```

Figure 69. Highlighted Phrase Example

Reference Phrases

The RP (reference phrase) tag allows you to highlight words or phrases on panels to indicate that additional help information is available. When a help panel with reference phrases is displayed, the cursor is positioned in the first reference phrase. When an application panel containing reference phrases is displayed, the cursor will be positioned to the first reference phrase or panel input field, unless the cursor setting has been specified by the application. The reference phrase is an input-capable field so that the user can tab to successive reference phrases on the panel. The reference phrase text is refreshed whenever the panel is displayed.

When the user places the cursor on a reference phrase and requests help, the reference phrase panel or message is displayed. Reference phrase help panels themselves can also contain reference phrases. When a user cancels a reference phrase help, the panel from which the user requested the reference phrase help is displayed again. All other help facilities, such as KEYSHELP and EXHELP, are available from a reference phrase help panel.

The RP tag requires a matching end tag to indicate the end of the reference phrase text.

The following markup example shows you how to use the RP tag.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=frenchl depth=12>Help for Masters Degree in French Literature
<area>
<info>
<p>
The Masters in French Literature (MFL) Program is also available
to students interested in
<rp help=liteve>evening studies.</rp>
<p>
Please consult your program advisers for details before registering for
a class.
</info>
</area>
</help>

<help name=liteve depth=13>Help for Evening Studies
<area>
<info>
<p>
Evening Studies offered by the French Literature
graduate program are available to students
interested in part-time and full-time studies.
All core courses and many electives are offered
in the evening on a rotating basis.  Please
consult your program advisers for details before
registering for a class.
</info>
</area>
</help>
```

Here's the formatted result:

```
  Help for Masters Degree in French Literature

The Masters in French Literature (MFL) Program
is also available to students interested in
evening studies.

Please consult your program advisers for details
before registering for a class.

 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 70. Reference Phrase Example

Accordingly, when the user selects the reference phrase **evening studies**, the help panel specified by the HELP attribute (help=liteve) is displayed.

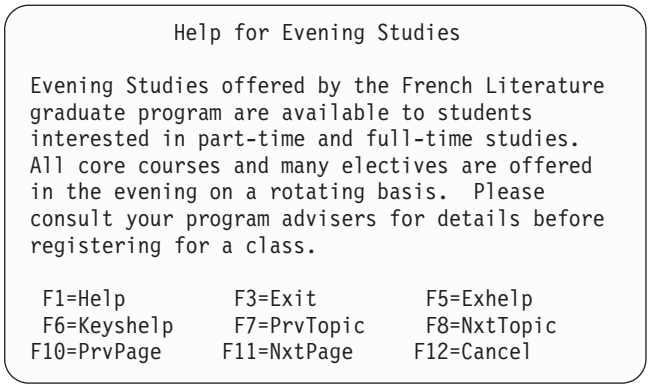


Figure 71. Reference Phrase Example of Help Attribute

The *help-panel-name* attribute specifies the name of the help panel to be displayed if the reference phrase is selected.

Using Information Regions with Other Panel Elements

You can use information regions to complement the other elements of an application panel in many different ways. For example, you can use an information region to provide additional information for fields on an application panel.

The information region in the following markup example uses a paragraph and a compact ordered list to tell the user how to interact with the panel fields. Figure 72 on page 149 shows the formatted result.

```
<!doctype dm system>

<VARCLASS NAME=selcls TYPE='char 1'>

<VARLIST>
  <VARDCL NAME=day VARCLASS=selcls>
  <VARDCL NAME=time VARCLASS=selcls>
</VARLIST>

<panel name=appmnt>Make an Appointment
  <area>
    <info width=74>
      <p>To schedule an appointment, you must choose one
      selection from each field.
      <ol compact>
        <li>Choose a day from the first field.
        <li>Choose a time slot from the second field.
        <li>After you have completed both fields, press
        Enter to log your appointment and leave the panel.
      </ol>
    </info>
    <divider type=solid gutter=3>
    <region dir=horiz>
      <region>
      <selfld name=day selwidth=20 pmtwidth=9>Weekdays:
        <choice>Monday
        <choice>Tuesday
        <choice>Wednesday
        <choice>Thursday
        <choice>Friday
      </selfld>
      </region>
      <divider gutter=8>
      <region>
```

```
        <selfld name=time selwidth=20 pmtwidth=5>Time:
          <choice>9:00
          <choice>10:00
          <choice>11:00
          <choice>12:00
          <choice>1:00
          <choice>2:00
          <choice>3:00
          <choice>4:00
        </selfld>
      </region>
    </region>
  </area>
</panel>
```

```
                          Make an Appointment

To schedule an appointment, you must choose one selection from each field.

1.  Choose a day from the first field.
2.  Choose a time slot from the second field.
3.  After you have completed both fields, press Enter to log your
    appointment and leave the panel.

 ------------------------------------------------------------------------

Weekdays:                   Time:
__  1.  Monday              __  1.  9:00
    2.  Tuesday                 2.  10:00
    3.  Wednesday               3.  11:00
    4.  Thursday                4.  12:00
    5.  Friday                  5.  1:00
                                6.  2:00
                                7.  3:00
                                8.  4:00
```

Figure 72. Information Region

Help Panels

In this section, we show you how to use the DTL to define help panels that provide help to users while they are using an ISPF application. We also show you how to link help panels with application panels.

Defining Help Panels

The HELP tag and its required end tag define a help panel. The HELP start tag indicates the beginning of a help panel definition, and the HELP end tag closes the definition. All of the other tags that compose a help panel are coded between these two tags. You also use the HELP tag to define the help panel title in the same way you code panel title text with the PANEL tag, as tag content.

The HELP tag and its matching end tag look like this:

```
<help name=help01>Help Panel Title
</help>
```

In the above example, we added the required NAME attribute and value to the HELP start tag. The NAME value you assign must follow the standard naming convention described in "Rules for Variable Names" on page 211.

The value you assign to NAME is the value that elements such as application panels, fields, and messages use to provide help to the user.

For example, if we define the help we want to provide for an application panel in a help panel with the NAME value *help01*, we would specify that help panel like this in the PANEL definition:

```
<panel name=panel01 width=60 depth=18 help=help01>Application
```

The help panel *help01* would appear when the user requests help for that application panel.

Like the PANEL tag, the HELP tag has WIDTH and DEPTH attributes that define the dimensions of the panel. However, help panels differ from application panels. If the DEPTH attribute is specified on the AREA tag, a single panel is created with a scrollable section to allow the display of longer sections of help text. Otherwise, the conversion utility generates as many help panels as needed (up to 37) for the help text content you define. This means that you can define text for a help panel that exceeds the defined depth, and, even though the text may not appear in the initial display of the panel, the user can view the text through page scrolling. Examples of both types of help panel scrolling are shown on page 152 and Figure 79 on page 156.

Because ISPF displays all DTL-defined help panels in pop-ups, the WIDTH and DEPTH values you specify must allow for the addition of two lines (depth) and 4 characters (width) for pop-up borders. Therefore, WIDTH=76 and DEPTH=22 are the maximum values that can be used with 80-by-24 display devices. The HELP panel default values are WIDTH=50 and DEPTH=10.

Typically, you would define help panel values of WIDTH=60 and DEPTH=22 or less. The specified depth must include allowance for the panel title line and its separator. A help panel that does not end with a scrollable area also reserves four lines for the function key area.

Defining Help Panel Text

The text you define for help panels cannot be modified by the user; it is for information purposes only. To define this text, use an information region and the tags associated with information regions. The INFO tag and its matching end tag are required in help panel definitions.

You can also use AREA definitions within help panels. Remember to code the entire INFO definition (start and end tag) within the AREA definition, just as you would on an application panel. Here's an example:

```
<help name=help01>Help Panel Title
  <area>
    <info>
⋮

    </info>
  </area>
</help>
```

You can use any of the information region tags discussed earlier in this chapter in a help panel. For example, you use the P (paragraph) tag to define a paragraph of text on a help panel the same way you use it to define a paragraph on an application panel.

In the following help panel markup, we use two paragraphs, an unordered list, a figure and figure caption to define the help text. The specification of DEPTH=28 is valid only if the display terminal has 30 or more display lines. Figure 73 on page 152 shows the formatted result.

```
<!doctype dm system>

<help name=olcthlp depth=28 width=50>Help for Online Catalog
<area>
<info>
<p>The Online Catalog provides
you with:
<ul compact>
<li>The book title
<li>Catalog number
<li>Page count
<li>The author
<li>A brief description
</ul>
<p>Here is an example:
<fig>
  The Yellow Subroutine
  365 Pages              1234.56
  John-Paul Georgenringo

  A young band of British programmers embarks on
  a voyage across a perilous "sea" language in
  search of FORTRAN and fame.
<figcap>Online Catalog Example
</fig>
</info>
</area>
</help>
```

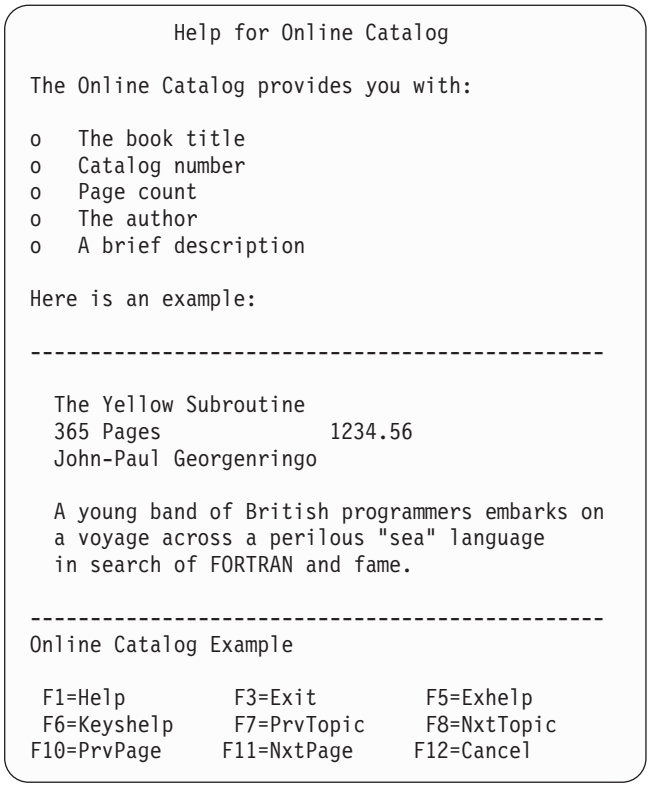


Figure 73. Help Panel

In Figure 73, all of the text was displayed because the depth we defined for the help panel was large enough to accommodate the text. However, the amount of help you want to provide for your users can vary, and it's not always possible to display all of the help text you define in the initial panel display, especially when you don't, or can't, specify a large DEPTH value for the help panel.

Depending on the use of the AREA tag, the conversion utility generates multiple panels or a single scrollable help panel.

The following help panel markup includes an information region that contains a paragraph, a definition list, and two unordered lists nested within the definition list.

The addition of the DEPTH attribute on the AREA tag illustrates a scrollable panel.

```
<!DOCTYPE DM SYSTEM>

<help name=helpscr width=46 depth=16>ShelfBrowse for Kids
<area depth=10>
  <info>
    <p>ShelfBrowse can help you
    find any kind of book you are looking for.
    The two main categories for books are:
    <dl tsize=12>
      <dthd>Book
      <ddhd>Description
      <dt>Fiction
      <dd>Fiction books are stories
      that never really happened.
      The writer made them up.
      For example:
```

```
        <ul>
          <li>Fairy Tales
          <li>Mysteries
          <li>Science fiction stories
        </ul>
      <dt>Nonfiction
      <dd>Nonfiction books are about
      things that really exist.
      For example:
        <ul>
          <li>History books
          <li>Reference books
          <li>How-to books
        </ul>
    </dl>
  </info>
</area>
</help>
```

When initially displayed, the first part of the scrollable text is visible. For this example, to scroll down, place the cursor on the first or last displayed line of text, and press Enter or the RIGHT (F11) key. Use the LEFT (F10) key to scroll up.

```
HELPSCR     ShelfBrowse for Kids

                                  More:     +

ShelfBrowse can help you find any kind of
book you are looking for. The two main
categories for books are:

Book         Description

Fiction      Fiction books are stories that
             never really happened.  The

 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 74. Help Panel (Example 1 of 4)

After scrolling down, the following panel appears:

```
HELPSCR      ShelfBrowse for Kids

                                More:   - +
             never really happened. The
             writer made them up. For
             example:

             o   Fairy tales

             o   Mysteries

             o   Science fiction stories

  F1=Help         F3=Exit         F5=Exhelp
  F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
 F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 75. Help Panel (Example 2 of 4)

After scrolling down, the following panel appears:

```
HELPSCR      ShelfBrowse for Kids

                                More:    - +
             o   Science Fiction stories

Nonfiction   Nonfiction books are about
             things that really exist. For
             example:

             o   History books

             o   Reference books

  F1=Help         F3=Exit         F5=Exhelp
  F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
 F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 76. Help Panel (Example 3 of 4)

After scrolling down, the following panel appears:

Note that because there is only one additional line to display, the scroll has moved the scrollable text up only one line.

```
HELPSCR     ShelfBrowse for Kids

                                  More:   -
Nonfiction  Nonfiction books are about
            things that really exist. For
            example:

            o   History books

            o   Reference books

            o   How-to books

 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 77. Help Panel (Example 4 of 4)

If no AREA tag is present or the AREA tag does not contain the DEPTH attribute, multiple help panels are generated. ISPF simulates scrolling by displaying the set of multiple help panels in sequence.

If the help panel contains additional text, the conversion utility provides an indicator at the top of the panel to notify the user. If additional text exists, the text **More:** is displayed followed by a + sign. Following scrolling, if additional text stills exists, the indicator displays as "**More: – +**", indicating scrolling is possible in either direction. If, following scrolling, no more text is available through scrolling forward, but text is available by scrolling backward, the indicator displays as "**More: –**". Scrolling function keys are defined by tutorial processing.

The following markup uses the previous example without a DEPTH attribute on the AREA tag to generate multiple help panels. Because all of the data does not fit in one help panel, the conversion utility created three panels: HELPSB, HELPSBX0, and HELPSBX1. The panels are displayed individually by tutorial processing. Figures 78, 79, and 80 show the formatted results with the function key area displayed in its short form.

```
<!DOCTYPE DM SYSTEM>

<help name=helpsb width=46 depth=16>ShelfBrowse for Kids
<area>
  <info>
    <p>ShelfBrowse can help you
    find any kind of book you are looking for.
    The two main categories for books are:
    <dl tsize=12>
      <dthd>Book
      <ddhd>Description
      <dt>Fiction
      <dd>Fiction books are stories
      that never really happened.
      The writer made them up.
      For example:
        <ul>
          <li>Fairy Tales
          <li>Mysteries
          <li>Science fiction stories
        </ul>
      <dt>Nonfiction
      <dd>Nonfiction books are about
```

```
      things that really exist.
      For example:
        <ul>
          <li>History books
          <li>Reference books
          <li>How-to books
        </ul>
    </dl>
  </info>
</area>
</help>
```

```
HELPSB      ShelfBrowse for Kids
                                  More:     +
ShelfBrowse can help you find any kind of
book you are looking for. The two main
categories for books are:

Book        Description

Fiction     Fiction books are stories that
            never really happened. The
            writer made them up. For
            example:

 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 78. Help Panel (Example 1 of 3)

```
HELPSBX0    ShelfBrowse for Kids
                                  More:   - +
            o   Fairy Tales

            o   Mysteries

            o   Science fiction stories

Nonfiction  Nonfiction books are about
            things that really exist. For
            example:


 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 79. Help Panel (Example 2 of 3)

```
HELPSBX1    ShelfBrowse for Kids
                                More:   -
            o   History books

            o   Reference books

            o   How-to books




 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 80. Help Panel (Example 3 of 3)

As we stated earlier, you can use any of the tags provided for information regions to define the text of the information regions in your help panels.

Chapter 7. Messages

You use messages to communicate information to users; that is, information that you, the application developer, believe they need to know. Typically, this would be information regarding user actions, status, or problems that need correction. Additionally, ISPF issues messages when needed to inform users of situations that ISPF handles.

The Dialog Tag Language provides you with the means of defining application-provided messages. You use ISPF services to handle the display of messages. When the application calls for a message to be displayed, ISPF places it either in the message area of the application panel or within a pop-up window, known as a *message pop-up*.

Messages are defined according to their purpose and severity. The four types of messages you can define are:

Information	To provide information about a user-requested action.
Warning	To provide information about conditions the user may need to be aware of.
Action	To alert the user to an exception condition that requires a response from the user to correct the situation.
Critical	To alert the user to an exception condition that requires a response from the user to correct the situation. Critical messages are similar to action messages.

This chapter tells you how to define messages for your applications. For a complete description of ISPF message processing, refer to *ISPF Dialog Developer's Guide and Reference*.

Defining Messages

The messages you define using DTL are stored within *message members*. Each regular message member can contain up to 10 messages. The conversion utility stores the message members in an ISPF message file for the application. The DTL tags you use to define messages and message members are:

MSGMBR
Defines a message member. The MSGMBR tag requires an end tag.

MSG Defines a message within a message member. The text of the MSG tag is the text that appears as the message. Each message can be up to 512 bytes in length after variable substitution. "Variable Substitution" on page 164 describes variable substitution in messages.

You assign an identifier to each message within a message member. The message identifier is composed of two required attribute values: the NAME attribute value of the MSGMBR tag and the SUFFIX attribute value for the MSG tag.

© Copyright IBM Corp. 1989, 2001

The NAME attribute you specify for the MSGMBR tag can consist of 1–5 uppercase or lowercase alphabetic characters and 2 numeric characters.

The SUFFIX attribute values for each of the MSG tags you code within a MSGMBR definition must consist of either 1 numeric character (0–9) or a numeric character (0–9) and an optional suffix character as defined for ISPF messages. Each of the values must be unique (a message suffix cannot be defined twice in a message member).

```
<!doctype DM system>

<msgmbr name=maia00>
  <msg suffix=0>You cannot type a number in the Name field.
  <msg suffix=1>Please include your first name in the Name field.
  <msg suffix=2>Unrecognized character in Name field.  Please correct.
  <msg suffix=3>Unrecognized character in Address field.  Please correct.
  <msg suffix=4>You cannot type a number in the City field.
  <msg suffix=5>Unrecognized character in City field.  Please correct.
  <msg suffix=6>You cannot type a number in the State field.
  <msg suffix=7>You must type two letters in the State field.
  <msg suffix=8>The Zip code exceeds the maximum length.
  <msg suffix=9>You cannot type an alphabetic character in the Zip field.
</msgmbr>
```

The value of the MSG SUFFIX attribute, when added to the MSGMBR NAME value, forms the message identifier for that message. For example, the message identifier for the message: "You must type two letters in the State field". is *maia007*. If you specify *maia007* as the MSG value on a CHECKL tag, this message is displayed when ISPF detects the error as a result of input validation.

In addition to SUFFIX, the MSG tag has an optional HELP attribute that allows you to identify a help panel for the message. For information on defining help panels, see "Chapter 6. Information Regions and Help Panels" on page 115.

Specifying Message Severity

The severity you assign a message determines if the alarm is sounded when the message is displayed. You can specify the severity of a message with the MSGTYPE attribute of the MSG tag. ISPF accepts one of four values for the MSGTYPE attribute: INFO (the default value), WARNING, ACTION, or CRITICAL. The value can be supplied as a variable name.

Information Messages

Use the default value INFO when you want to provide the user with feedback about the state of the application.

```
<msgmbr name=orda00>
  <msg suffix=0>Your order is being processed.  Please wait...
</msgmbr>
```

Warning Messages

Warning messages tell users that a potentially undesirable situation could occur. Users only need to respond to the message to continue, although corrective action may be required later. ISPF displays warning messages with an alarm.

```
<msgmbr name=orda00>
  <msg suffix=0>Your order is being processed.  Please wait...
```

```
  <msg suffix=1 msgtype=warning>Your request for the engraving
  option is not valid.
  Please check your request, and correct it if necessary.
</msgmbr>
```

Action and Critical Messages

Action and critical messages both represent the highest degree of severity. They tell users about exception conditions that require a response. The user must respond with a specific action to continue with the application. ISPF displays these messages with an alarm.

Action messages may appear in a pop-up or in the panel message area. Critical messages always appear in a pop-up.

```
<!doctype dm system>

<msgmbr name=orda00>
  <msg suffix=0>Your order is being received.  Please wait...
  <msg suffix=1 msgtype=warning>Your request for
  the engraving option is not valid.
  Please check your request, and correct it if necessary.
  <msg suffix=2 msgtype=action>The data you have
  entered is incorrect.
  Please reenter the data.
</msgmbr>
```

Short Messages

The SMSG attribute enables you to specify a short message. The short message does not conform to CUA architecture, but it is supported for ISPF compatibility.

Assigning Messages

Some of the DTL tags have an optional MSG attribute that you use to specify a message-identifier. The message text associated with the message-identifier specified is displayed when conditions you define for the tag are not met by the user.

The following list contains the DTL tags that have MSG attributes associated with them, and describes the conditions for each.

CHECKL
Use the MSG attribute of the CHECKL tag to specify a message ISPF displays when the user's input fails the validation check defined for the check list.

CHOFLD
Use the MSG attribute of the CHOFLD tag to specify a message ISPF displays when the user does not provide input for a required field. You can only assign a message to a data field when the REQUIRED attribute has a value of YES.

DTAFLD
Use the MSG attribute of the DTAFLD tag to specify a message ISPF displays when the user does not provide input for a required field. You can only assign a message to a data field when the REQUIRED attribute has a value of YES.

LSTCOL
Use the MSG attribute of the LSTCOL tag to specify a message ISPF displays when the user does not provide input for a required entry. You can only assign a message to a list column when the REQUIRED attribute has a value of YES.

SELFLD

Use the MSG attribute of the SELFLD tag to specify a message ISPF displays when the user does not provide input for a required single-choice selection field. You can only assign a message to a selection field when the REQUIRED attribute has a value of YES.

Use the SELMSG attribute of the SELFLD tag to specify a message ISPF displays when the user selects an invalid choice.

Use the SELMSGU attribute of the SELFLD tag to specify a message ISPF displays when the user selects an unavailable choice.

VARCLASS

Use the MSG attribute of the VARCLASS tag to specify a message ISPF displays when the user's input fails the validity check defined by the VARCLASS TYPE attribute.

Note: The message specified by the MSG attribute of a VARCLASS tag is also used if enclosed checks (CHECKL tag) or translations (XLATL tag) do not include the MSG attribute.

XLATL

Use the MSG attribute of the XLATL tag to specify a message that ISPF displays when the user's input fails a specified translation.

ISPF displays a default message for most of the situations listed above if you do not specify the MSG attribute.

To show you how messages are associated with DTL tags, the following example defines a data field that requires input from the user. It also defines a message member that contains the warning message ISPF displays if the user does not provide input for the field. Figure 81 on page 163 shows the displayed panel and message.

```
<!doctype dm system>

<varclass name=namecls type='char 30'>

<varlist>
  <vardcl name=name varclass=namecls>
</varlist>

<msgmbr name=ordb00>
  <msg suffix=0 msgtype=warning>You must type your name in the Name field.
</msgmbr>

<panel name=msgxmp1>Application Panel
 <dtafld datavar=name pmtwidth=12 required=yes msg=ordb000>Name
<cmdarea>
</panel>
```

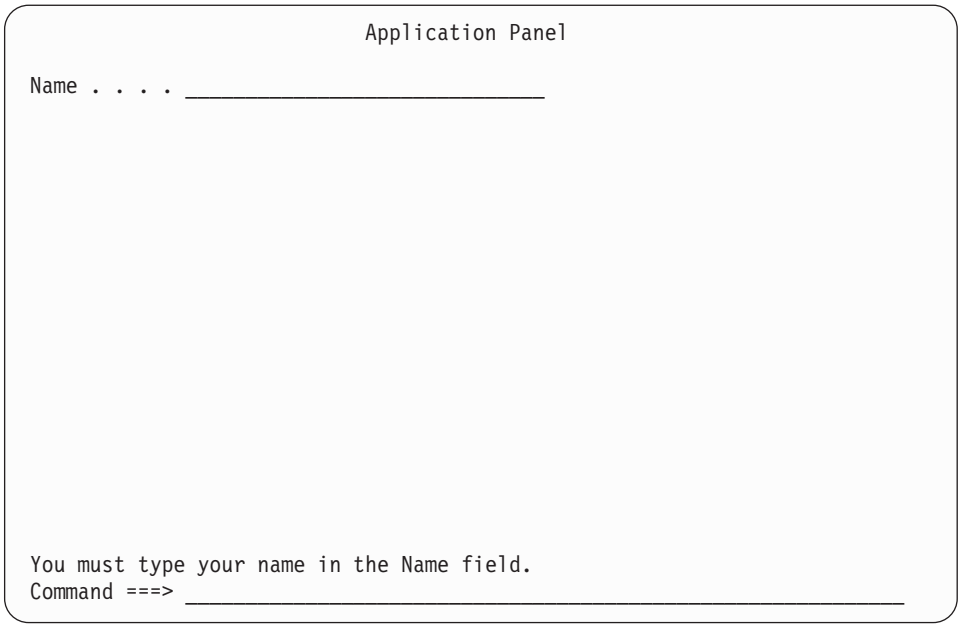


Figure 81. Data Field and Message

Displaying Messages

You can specify how a message will be displayed, either in the panel message area or a pop-up, using the LOCATION attribute of the MSG tag.

There are five valid values you can assign to LOCATION: AREA (the default), MODAL, MODAL(L), MODELESS, and MODELESS(L). AREA specifies that the message is to appear in the panel message area, unless the text of the message exceeds the length of the message area. If the text of the message exceeds the message area length, ISPF displays the message in a pop-up.

If you want a message that requires a response from the user to appear in a pop-up, specify the MODAL or MODAL(L) value for the LOCATION attribute. This is useful for presenting warning and action messages that have a good deal of text.

If you want a message that does not require a response from the user to appear in a pop-up, specify the MODELESS or MODELESS(L) value for the LOCATION attribute.

For further discussion of these LOCATION values, see “MSG (Message)” on page 393.

The following message member markup contains three messages, each of them with a different LOCATION value. The second and third messages display in pop-up windows.

```
<!doctype dm system>
<msgmbr name=orda01>
  <msg suffix=0>Your order is being received.  Please wait...
  <msg suffix=1 msgtype=warning location=modeless>Your request for
  the engraving option is not valid.
```

```
  Please check your request, and correct it if necessary.
  <msg suffix=2 msgtype=action location=modal>The data you have
  entered is incorrect.
  Please reenter the data.
</msgmbr>
```

Variable Substitution

You can specify a variable in the text of a message by using the VARSUB (variable substitution) tag. When the message is displayed, ISPF inserts the current value of the variable into the text of the displayed message.

You code the VARSUB tag within the text of the message where you want the substitution to be made. You use the required VAR attribute of the VARSUB tag to specify the name of a declared variable whose value is substituted in the message text.

In the following example, we used two variable substitutions in the text of the message "msga001". The first VARSUB specifies the variable *invvar*, which provides an invoice number in the message. The second VARSUB specifies the variable *datevar*, which provides a date in the message.

```
<!doctype dm system>

<varclass name=invoices type='char 10'>
<varclass name=updates  type='char 8'>

<varlist>
  <vardcl name=invvar varclass=invoices>
  <vardcl name=datevar varclass=updates>
</varlist>

<msgmbr name=msga00>
  <msg suffix=0>Your request is being processed.
  <msg suffix=1>The invoice number you have requested,
  <varsub var=invvar>, was last updated on
  <varsub var=datevar>.
</msgmbr>
```

Chapter 8. The Application Command Table

In addition to the commands in the ISPF system command table, DTL provides you with a way to define and store commands that are specific to your application. You can also define commands that override the ISPF system commands. You define and store these commands within a *command table* for your application. These application-specific commands define the responses to commands entered by the user in the command entry field and commands linked to pull-down choices and key mapping lists.

You can define only one command table for an application. ISPF locates the command table using the defined *application-identifier* for the command table.

For a complete description of ISPF command processing and a list of the ISPF system commands, refer to the *ISPF User's Guide*

Note: You can use the TSO ISPCMDTB command to convert existing command tables to DTL. To use ISPCMDTB, ensure that the command table is in your table concatenation (ISPCMDTB), type **TSO ISPCMDTB** *applid* (where applid is the application id of the command table). This will place you in an edit session containing the DTL version of the command table. Use the editor CREATE or REPLACE commands to save the table to your DTL source data set.

Defining the Application Command Table

The tags you use to define an application command table are:

CMDTBL Begins the definition of an application command table. The required end tag ends the definition.

CMD Defines a command within an application command table. You code the CMD tags within a CMDTBL definition (between the start and end tags).

CMDACT Defines the action taken by ISPF when a user enters a command. You code the CMDACT tag following the command (CMD) with which it is associated.

The CMDTBL tag has a required APPLID attribute that you use to define the *application identifier* for the command table. ISPF uses the value you assign with the APPLID attribute to identify the command table. The value you assign to APPLID must be the same as the run-time application identifier specified when the application starts.

The value you assign as an application identifier can have a maximum of 4 characters, and the first character must be A-Z, a-z, @, #, or $.

Any remaining characters can be either A-Z, a-z, @, #, $ or 0-9. Lowercase characters are translated to their uppercase equivalents. Additionally, ISPF reserves the application identifier ISPx, where x is any character including the space character. Do not use any of these for an APPLID value.

© Copyright IBM Corp. 1989, 2001

The conversion utility uses the application identifier as a prefix to the string CMDS to form the name of the command table library. For example, the APPLID value, *demo*, results in the application command table name DEMOCMDS.

Command tables are updated using ISPF table services. Input is obtained from the ISPTLIB DDname allocation and output is written to the ISPTABL DDname allocation. Refer to the description of how to allocate libraries before starting ISPF in the *ISPF User's Guide* for more information about the use of ISPTLIB and ISPTABL.

When a user enters a command in a command-entry field or through a pull-down choice or function key, ISPF searches the command tables defined for the user. Any or all of the following tables will be searched (in the order shown below) if the table is present and defined. (A user and site command table can be defined in the ISPF Configuration table and the search order of the site and system command table can be reversed if specified as such in the ISPF Configuration table.)

1. Application command table
2. User command table
3. Site command table
4. System command table

If the command is found in a command table, ISPF performs the action defined in that command table for that command. If the command is not found in any of the command tables, ISPF passes the command to the application program for processing. If any of the command tables are not present, ISPF skips to the next command table in the hierarchy.

Use the CMD tag to define each of the commands within the application command table. The CMD tag has a required NAME attribute that you use to identify the *internal-command-name* for the command. The value you assign as an *internal-command-name* must not exceed 8 characters, and the first character must be alphabetic. Any remaining characters can be either alphabetic or numeric.

The following markup example shows a source file that contains an application command table, a key mapping list, and a panel with an action bar. The command table contains commands that are mapped to the RUN attributes of the ACTION tags associated with the pull-down choices and to the CMD attributes of the KEYI tags.

```
<!doctype dm system>

<cmdtbl applid=brws>
  <cmd name=quit>quit
    <cmdact action=...>
  <cmd name=send>send
    <cmdact action=...>
</cmdtbl>

<keyl name=panlkeys>
  <keyi key=f4 cmd=quit>
  <keyi key=f6 cmd=send>
</keyl>

<panel...>
  <ab>
    <abc>Actions
      <pdc>Quit
        <action run=quit>
      <pdc>Send
```

```
        <action run=send>
      <pdc>Exit
        <action run=exit>
  </ab>
⋮

</panel>
```

Because ISPF provides the EXIT command, it is not defined within the application command table. When the EXIT command is entered, ISPF finds it in the system command table.

Specifying Command Actions

You must specify a CMDACT tag for each of the CMD tags you define within an application command table so that ISPF can process these commands. You use the CMDACT tag to define the action taken for the command. Code the CMDACT tag immediately after the CMD tag it is associated with.

The ACTION Attribute

The CMDACT tag has a required attribute, ACTION, which you use to specify the ISPF command action. You can assign one of the ISPF command actions in the following list and some of the ISPF-provided system commands as listed in "CMDACT (Command Action)" on page 270. You can also specify command actions dynamically at run time as discussed in "Specifying Command Actions Dynamically" on page 168.

ALIAS To allow a command to have an alternate name, such as using QUIT as an alias for EXIT.

PASSTHRU To pass the command to the application. The *internal-command-name* and any command parameters are passed to the dialog in the ISPF ZCMD system variable.

SETVERB To pass the command to the application. The *internal-command-name* is passed to the dialog in the ZVERB system variable, and the parameters (if any) are passed to the dialog in the ZCMD system variable.

The ALIAS command action provides you with a way to define synonyms for commands. The *internal-command-name* you define for the ALIAS attribute value defines the command to be processed. You must enclose the keyword ALIAS, the *internal-command-name*, and any optional parameters within quotes.

When you define an ALIAS command action, you must code that command's CMD and CMDACT tags before the command the ALIAS represents. ISPF searches the application-defined commands first, and then searches the ISPF system commands. It must locate the ALIAS definition before the aliased command.

In the following example, we've added the commands PREV and NEXT to the application command table. We want "PREV" and "NEXT" to be aliases for the ISPF system commands BACKWARD and FORWARD. Because the BACKWARD and FORWARD commands are provided by ISPF, we do not need to define them in the application command table. ISPF locates the aliases before the ISPF system commands they refer to.

Additionally, this example shows the CMDACT for the SEND command set to PASSTHRU, because we want the application program to process the SEND command.

```
<cmdtbl applid=brws>
  <cmd name=quit>quit
    <cmdact action='alias exit'>
  <cmd name=send>send
    <cmdact action=passthru>
  <cmd name=prev>
    <cmdact action='alias backward'>
  <cmd name=next>
    <cmdact action='alias forward'>
</cmdtbl>
```

Specifying Command Actions Dynamically

You can also specify a variable as the value for the ACTION attribute of the CMDACT tag. ISPF substitutes the value of the variable at run time when the command is processed. The run-time value of the variable must be one of the ISPF-supported command actions. You specify the variable using the % notation in the ACTION value.

In the following example, we specified the variable *scroll* as a command action for the SCROLL command. When the user issues the SCROLL command, ISPF obtains the value of the variable *scroll* from the variable pool to determine the action to be taken. The application can then control the direction of scrolling by setting the variable *scroll* to FORWARD or BACKWARD, or to NOP if no scrolling is possible.

```
<!doctype dm system>

<cmdtbl applid=abcd>
  <cmd name=scroll>scroll
    <cmdact action='%scroll'>
⋮

</cmdtbl>
```

Truncating Commands

Using the command facilities as we have discussed them so far in this chapter, the user must enter a full command name when typing a command in the command area. You can provide a shortcut for the user by defining *command truncations* for commands. The user can issue a truncated command in the command area by entering the minimum number of characters you specify for the command.

To specify truncation for a command, you code the T (truncation) tag within the *external-command-name* of the command.

For example, to specify "qu" as the minimum command for the QUIT command, you add the T tag to the *external-command-name*, like this:

```
<cmdtbl applid=brws>
  <cmd name=quit>qu<t>it
    <cmdact action='alias exit'>
⋮

</cmdtbl>
```

The T tag follows the characters you specify as the minimum command.

With this truncation, the user can issue the QUIT command by typing the command in one of the following ways:

```
qu
qui
quit
```

However, you should be careful to avoid adding truncations that duplicate other truncations in the command table. For example, these two truncations define minimum commands (“co”) that are identical:

```
<cmdtbl applid=brws>
  <cmd name=comp>co<t>mpare
    <cmdact action=passthru>
  <cmd name=copy>co<t>py
    <cmdact action=passthru>
</cmdtbl>
```

The preceding definition would cause the conversion utility to issue a warning message.

To avoid this type of duplication, place the T tag appropriately in the CMD tag content. The above duplication can be avoided by coding the truncations as follows:

```
<cmdtbl applid=brws>
  <cmd name=comp>com<t>pare
    <cmdact action=passthru>
  <cmd name=copy>cop<t>y
    <cmdact action=passthru>
</cmdtbl>
```

Chapter 9. Defining Key Mapping Lists

Every application panel has keys that map to valid actions for the panel. You define these key assignments within key mapping lists. The key assignments map a key to a command defined within the application command table or to an ISPF-provided command. You use the KEYLIST attribute of the HELP, HELPDEF, PANEL or PANDEF tags to name the key mapping list to use for a panel. If a keylist is not specified, ISPF provides the default key mapping list used for help panels. ISPF also provides a default key mapping list used when application panels do not refer to an application-defined KEYLIST.

The tags you use to define key mapping lists are:

KEYL To define a key mapping list. The required end tag ends the key mapping list definition.

KEYI To define a key assignment and specify the command ISPF processes when the user presses the key, and specify the label for the key if it is displayed in the function key area.

You can code multiple KEYI (key item) tags within a KEYL (key list) definition. You code a KEYI tag for each key that is defined for the key mapping list.

Keylists are updated using ISPF table services. Input is obtained from the ISPTLIB DDname allocation and output is written to the ISPTABL DDname allocation. Refer to the description of how to allocate libraries before starting ISPF in the *ISPF User's Guide* for more information about the use of ISPTLIB and ISPTABL.

Assigning Keys and Actions

The KEYL tag starts a key mapping list definition and provides the name of the key mapping list. You specify the key mapping list to be used with the KEYLIST attribute of the HELP, HELPDEF, PANEL, or PANDEF tag.

Each KEYI definition within a key mapping list maps a key assignment with a command. The command can be defined in the application command table, the user command table, the site command table, the system command table, or it can be one of the ISPF-provided commands. The required KEY and CMD attributes of the KEYI tag match the key with the command.

The KEYI definition in this example maps the F2 key on the user's keyboard with the SEARCH command in the application command table.

```
<!doctype dm system>

<cmdtbl applid=abcd>
 <cmd name=search>Search
   <cmdact action=passthru>
</cmdtbl>

<keyl name=panlkeys>
  <keyi key=f2 cmd=search>Search
</keyl>

<panel name=panl01 keylist=panlkeys>
```

© Copyright IBM Corp. 1989, 2001

```
⋮

</panel>
```

When the user presses the F2 key during the display of an application panel that refers to this key mapping list, ISPF processes the SEARCH command.

ISPF Default Key List

ISPF provides a default key mapping list named ISPKYLST for application panels. If you do not specify a key mapping list to be associated with a panel (using the KEYLIST attribute of the PANEL or PANDEF tag), ISPF uses the keys defined for ISPKYLST to display in the function key area of the panel when it is displayed. See "PANEL (Panel)" on page 415 for information about coding the PANEL tag.

The key mappings for ISPKYLST are:

Key	Command
F1	HELP
F2	SPLIT
F3	EXIT
F9	SWAP
F12	CANCEL
F13	HELP
F14	SPLIT
F15	EXIT
F21	SWAP
F24	CANCEL

ISPF provides a default key mapping list named ISPHELP for help panels. If you do not specify a key mapping list to be associated with a panel (using the KEYLIST attribute of the HELP or HELPDEF tag), ISPF uses the keys defined for ISPHELP to display in the function key area of the panel when it is displayed. See "HELP (Help Panel)" on page 340 for information about coding the HELP tag and Table 4 on page 345 for key mappings of the ISPHELP keylist.

You can override the ISPF default key mapping list by specifying a KEYLIST attribute in the panel definition. All keys that you want to be active, including those for ISPF-provided commands, must be specified in the key mapping list referred to by the KEYLIST attribute.

Displaying Keys

While all of the key assignments you define in a key mapping list are valid for the application panels that refer to the list, they only appear in the function key area (FKA) of the panel under the following conditions:

- You specify that the key is to be displayed by including FKA=YES in the KEYI tag, and
- The user has not turned off display of the function key area.

You use the FKA attribute of the KEYI tag to specify whether the key is to appear in the panel's function key area. The default FKA value, NO, means that the key will not appear. You must specify FKA=YES for the key to be displayed in the function key area.

When function keys are displayed in the function key area, the key you assign is displayed followed by an equal sign and the FKA text defined for the KEYI tag.

Defining Help for Key List

The conversion utility supports a keys help panel name on the KEYL tag. This allows a keys help panel to be associated with the key list. You can use the KEYLIST utility to add, change, or delete a keylist help panel name.

Alternatively, the application can provide the help panel name in the ZKEYHELP variable. However, the panel name specified as the keylist help panel either on the KEYL tag or by the KEYLIST utility overrides the panel name supplied by the ZKEYHELP variable.

In the following example, we want only the F2, F3, and F6 keys to appear in the panel function key area, with F2 mapped to the SEARCH command defined in the application command table, F3 mapped to the EXIT command, and F6 mapped to the KEYSHELP command. We also want F1 to be active to support the ISPF HELP command. No other function keys are to be active for this key mapping list. To obtain this result, we define the function key mapping list as follows:

```
<!doctype dm system>

<cmdtbl applid=abcd>
  <cmd name=search>Search
    <cmdact action=passthru>
</cmdtbl>
<keyl name=panlkeys help=pnlkeyh>
  <keyi key=f1 cmd=help>
  <keyi key=f2 cmd=search fka=yes>Search
  <keyi key=f3 cmd=exit fka=yes>Exit
  <keyi key=f6 cmd=keyshelp fka=yes>Keyshelp
</keyl>

<panel name=panl01 keylist=panlkeys>
⋮

</panel>
```

Here's how the function key area appears when panel "panl01" is displayed:

Figure 82. Displayed Function Key Area

Chapter 10. Using the Conversion Utility

The ISPF conversion utility is a tool that converts Dialog Tag Language (DTL) source files into ISPF panel language source format or executable preprocessed ISPF format. There are two methods of invoking the conversion utility–using the ISPF-supplied invocation panels, or using the conversion utility syntax. In either case, the conversion utility must be run under ISPF control. In this chapter, we explain both methods of calling the conversion utility.

Using the ISPF-Supplied Invocation Panels

Type the following command on the command line to invoke the conversion utility and display the ISPF invocation panel:

```
ISPDTLC
```

Invocation Panel

The following panel appears:

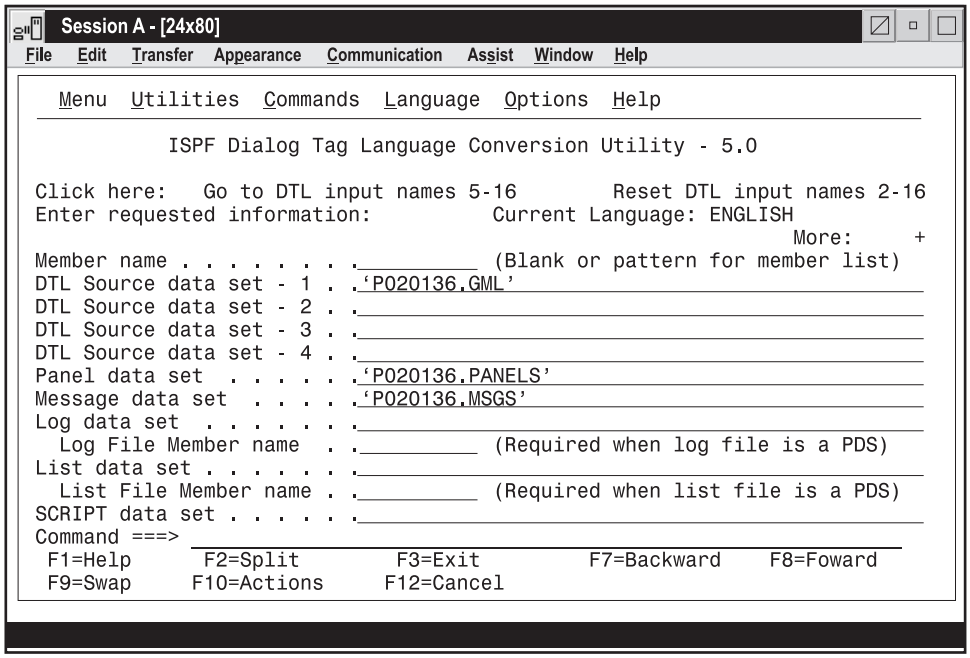


Figure 83. Conversion Utility Invocation Panel (ISPCP01)

When you scroll forward once, the following panel view appears.

© Copyright IBM Corp. 1989, 2001

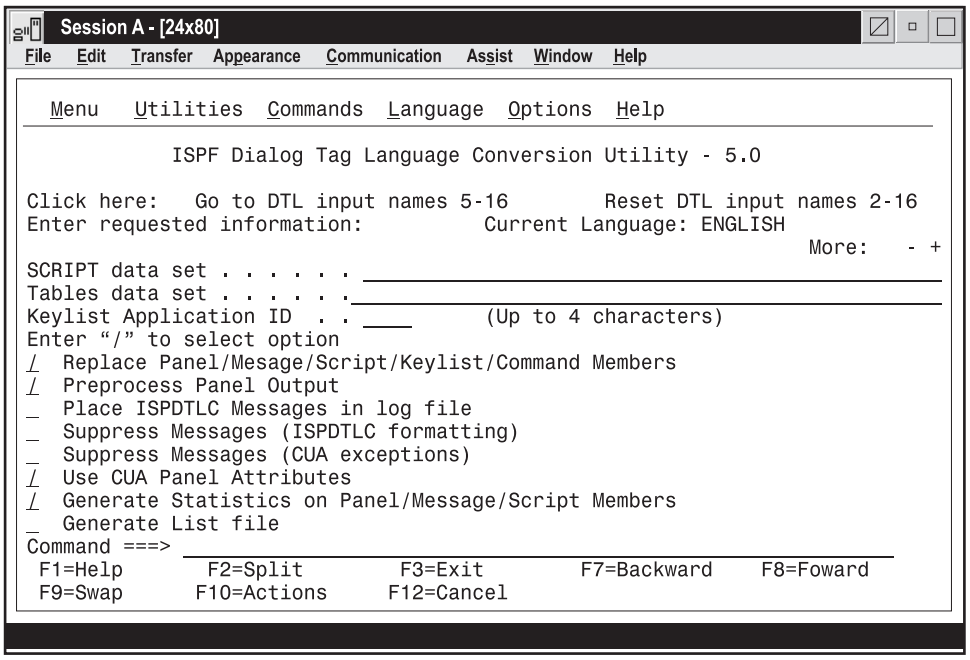


Figure 84. Conversion Utility Invocation Panel (Middle)

When you scroll forward a second time, the following panel view appears.

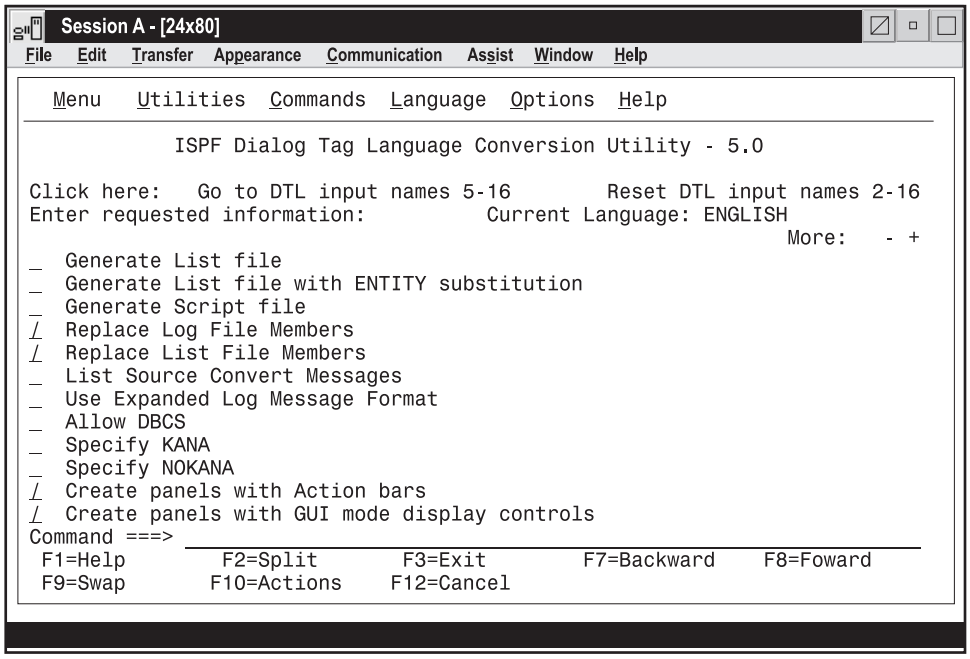


Figure 85. Conversion Utility Invocation Panel

When you scroll forward again, the following panel view appears.

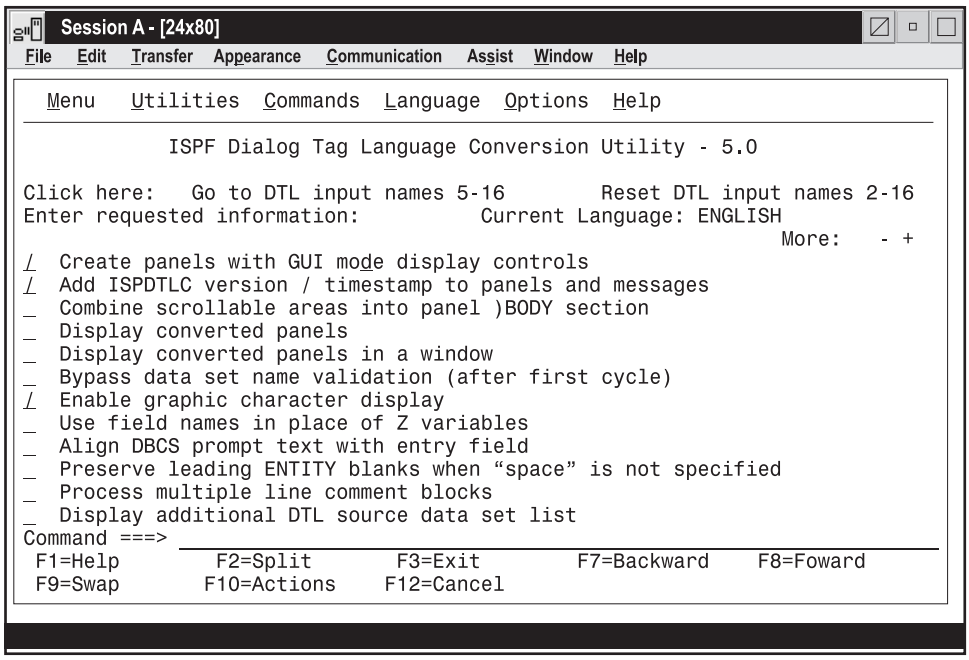


Figure 86. Conversion Utility Invocation Panel

Finally, when you scroll forward again, the following panel view appears.

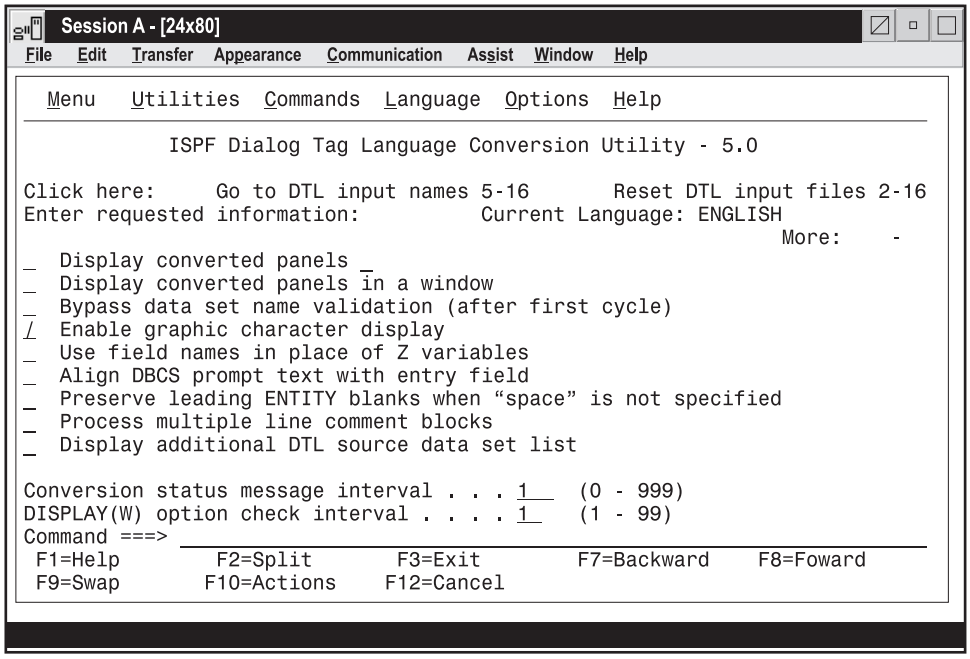


Figure 87. Conversion Utility Invocation Panel (End)

You must specify:
- At least one file containing DTL source
- The panel output file
- The message output file

The language selection defaults to the current ISPF session language. The current selected language is displayed as an information field on the panel.

Select the national language you want by using the “Language” action bar pull-down to enter a number corresponding to the supported ISPF language. The language is used to provide formatting rules for tag text. See “Text Formatting” on page 13 for more information.

Panel Input Fields

Additional information about the panel input fields follows:

Member Name
If the member name is left blank or entered as a member pattern, a member list is displayed. You can select one or more members to be converted from the member list.

DTL Source data set - n
You can specify up to three additional DTL source libraries on the invocation panel. See “Additional DTL Source Files” on page 181 for more information.

Panel data set
If no panel output is required, you can specify NULLFILE or DUMMY in place of the panel output file name.

Message data set
if no message output is required, you can specify NULLFILE or DUMMY in place of the message output file name.

Log data set
The log file name is optional. If it is not specified and the messages are to be written to disk, log output is written to the ISPF log file. If the log file is a PDS, a member name must be provided. You may specify an asterisk to tell the conversion utility to use the input GML source file member name as the output log file member name. However, if the input GML member is in the special DTLLST file list format (discussed in “Conversion Utility General Information” on page 189) then a separate log file member is created for each source member converted.

List data set
The list file name is optional. If it is not specified, list output is written to the ISPF list file. If the list file is a PDS, a member name must be provided. You may specify an asterisk to tell the conversion utility to use the input DTL source file member name as the output list file member name. However, if the input GML member is in the special DTLLST file list format (discussed in “Conversion Utility General Information” on page 189) then a separate list file member is created for each source member converted.

SCRIPT data set
The SCRIPT file name is optional. If a SCRIPT output file is requested, it must be a PDS file. Member names for the SCRIPT file are the same as the panel file.

Tables data set
The Tables file name is optional. If a tables file name is provided, it must be an 80 byte fixed length PDS file. When a tables file is provided, keylist and command table output is placed in this file.

Keylist Application ID
The optional Keylist Application ID is used when the APPLID attribute is omitted on the HELP, PANEL, KEYL, or CMDTBL tags. This is the equivalent of the ID provided by the KEYAPPL option described in “Conversion Utility Syntax” on page 183.

Conversion status message interval

When the conversion utility is running in interactive mode and the "Place ISPDTLC Messages in log file" option is selected and the "List Source Convert Messages" option is deselected, a status message containing the name of the current DTL source file member being converted is displayed in the long message area. This message provides a conversion status when you are converting multiple members using the DTLLST format member option. See page 192 for more information about the DTLLST syntax. The default message interval value is 1 which displays the message for each member processed. This value can be set to 0 to suppress the message (which is useful when running in GUI mode) or to a value that refreshes the message after a specified number of members have been converted.

DISPLAY(W) option check interval

When the conversion utility is running in test mode and either the DISPLAY or DISPLAYW option is selected, the converted panel is displayed for visual verification. A panel is displayed periodically after the converted panel has been displayed to enable the user to control the DISPLAY or DISPLAYW function.

The *DISPLAY(W) option check interval* option on the invocation panel controls the frequency of the DISPLAY or DISPLAYW control function panel appearance. The default value is 1, so that the control function panel is displayed after each converted panel display. The control panel enables you to continue using the same display interval, cancel the DISPLAY or DISPLAYW option, or change the control panel display interval.

All files specified must be pre-allocated.

When the log or list file is a PDS file and the member name is not an asterisk, all of the conversion results are placed in the specified member. If the file name or member name is changed, the pending log or list information is written to the previously specified member and a new log or list is generated beginning with the next conversion. When the conversion utility ends, pending log and list files are written.

The log and list files can be either fixed or variable length, with or without printer control. When the file is allocated with print control specified, the conversion utility output begins in column 2; column 1 is blank. When print control is not specified, the conversion utility output begins in column 1.

Panel Options

The conversion utility options are displayed either as a multi-choice selection list by scrolling the invocation panel, or in a series of multi-choice selection list panels with related options, through the Options pull-down on the action bar.

You select the options by entering a "/" in front of the option description. If you want to deselect an option, you must leave the selection choice field blank. These options are initially set to the default values described in "Conversion Utility Syntax" on page 183. The options and their valid values are equivalent to conversion utility syntax in the following manner. Note that *b* represents a blank.

Options	Valid Values
Replace Panel/Message/SCRIPT/Keylist/ Command Members	/ is equivalent to REPLACE, the default. *b* is equivalent to NOREPLACE.

Options	Valid Values
Preprocess Panel Output	/ is equivalent to PREP, the default. *b* is equivalent to NOPREP.
Place ISPDTLC Messages in log file	*b* is equivalent to SCREEN, the default. / is equivalent to DISK.
Suppress Messages (ISPF extensions)	*b* is the equivalent to NOMSGSUPP, the default. / is equivalent to MSGSUPP.
Suppress Messages (CUA exceptions)	*b* is equivalent to NOCUASUPP, the default, / is equivalent to CUASUPP.
Use CUA Panel Attributes	/ is equivalent to CUAATTR, the default. *b* is equivalent to NOCUAATTR.
Generate Statistics on Panel/Message/ Script Members	/ is equivalent to STATS, the default. *b* is equivalent to NOSTATS.
Generate List file	*b* is equivalent to NOLISTING, the default. / is equivalent to LISTING.
Generate List file with substitution	*b* is equivalent to NOFORMAT, the default. / is equivalent to FORMAT.
Generate SCRIPT file	*b* is equivalent to NOSCRIPT, the default. / is equivalent to SCRIPT.
Replace Log File Members	/ is equivalent to LOGREPL, the default. *b* is equivalent to NOLOGREPL.
Replace List File Members	/ is equivalent to LISTREPL, the default. *b* is equivalent to NOLISTREPL.
List Source Convert Msgs	*b* is equivalent to NOLSTVIEW, the default. / is equivalent to LSTVIEW.
Use Expanded Message Format	*b* is equivalent to NOMSGEXPAND, the default. / is equivalent to MSGEXPAND.
Allow DBCS	*b* is equivalent to NODBCS, the default. / is equivalent to DBCS.
Specify KANA	/ is equivalent to KANA.
Specify NOKANA	/ is equivalent to NOKANA.
Create panels with Action bars	/ is equivalent to ACTBAR, the default. *b* is equivalent to NOACTBAR
Create panels with GUI mode display controls	/ is equivalent to GUI, the default. *b* is equivalent to NOGUI
Add ISPDTLC version / timestamp to panel	/ is equivalent to VERSION, the default. *b* is equivalent to NOVERSION
Combine scrollable areas into panel body	*b* is equivalent to NOMERGESAREA, the default. / is equivalent to MERGESAREA.
Display converted panels (*)	*b* is equivalent to NODISPLAY, the default. / is equivalent to DISPLAY.
Display converted panels in a window (*)	*b* is equivalent to NODISPLAYW, the default. / is equivalent to DISPLAYW.

Options	Valid Values
Bypass data set name validation (after first cycle).	*b* is equivalent to DSNCHK, the default. */* is equivalent to NODSNCHK.
Enable graphic character display	*/* is equivalent to GRAPHIC, the default. *b* is equivalent to NOGRAPHIC.
Use full names in place of Z variables	*b* is equivalent to ZVARS, the default. */* is equivalent to NOZVARS.
Align DBCS prompt text with entry field	*b* is equivalent to NODBALIGN, the default. */* is equivalent to DBALIGN.
Preserve leading blanks when *space* is not specified	*b* is equivalent to NOPLEB, the default. */* is equivalent to PLEB.
Process multiple line comment blocks	*b* is equivalent to NOMCOMMENT, the default. */* is equivalent to MCOMMENT.
Display additional DTL source data set list	*b* —second input panel is not displayed. */* —second input panel is displayed.
(*): If you specify DISPLAY or DISPLAYW, ISPDTLC must be run in test mode (Option 7) to force display processing to use the current generated panel. An error message is issued if ISPDTLC is not being run in test mode and either option is specified.	

All of the entries from the panel (or panels) are saved in the user's profile.

Additional DTL Source Files

A second input panel is displayed for entry of up to twelve additional DTL source file data set names when any of the following occurs:

- you place the cursor on the point-and-shoot panel phrase **DTL input files 5-16** and press Enter.
- the **Display additional DTL source data set list** option is selected from either the scrollable area of the main panel or the *Miscellaneous* section of the **Options** action bar pull-down
- **XDTL** is entered on the command line
- Option 7 is selected from the **Commands** action bar pull-down

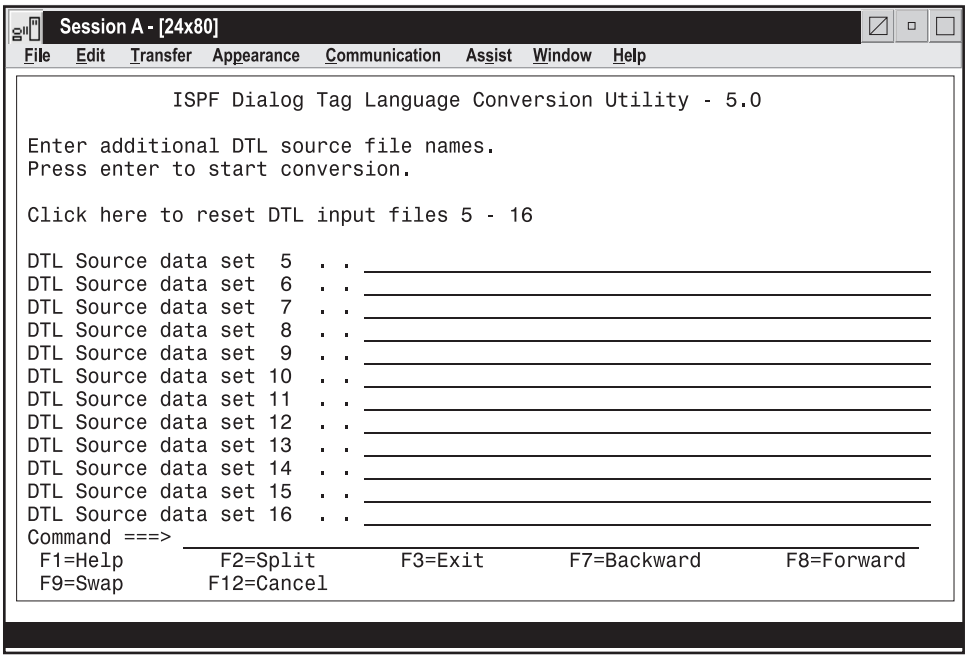


Figure 88. Panel ISPCP04

The entries for DTL source data sets 2-16 can be reset from the first invocation panel by placing the cursor on the point-and-shoot field **Click here to reset DTL input files 2-16** and pressing Enter. The panel redisplays with all entries except **DTL Source data set-1** reset to blanks.

Similarly, DTL source data sets 5-16 can be reset from the additional source files panel by placing the cursor on the point-and-shoot field **Click here to reset DTL input files 5-16** and pressing Enter. The invocation panel redisplays with all entries on the additional source files panel reset to blanks.

Converting Multiple DTL Source Files

When ISPF finishes processing the DTL source you specify, the conversion utility displays the invocation panel again to convert another DTL source file. This cycle continues until you exit or cancel the conversion utility.

Calling Help

You can get help for any field on the conversion utility invocation panel by moving the cursor to the field and pressing the F1 key.

Using CUA Panel Attributes

CUA defines the default colors and emphasis techniques for individual panel elements. The conversion utility generates panel element attributes that ISPF defines. The NOCUAATTR option can be used to create panels with attributes compatible with ISPF Version 3.1 and Version 3.2.

Refer to the *ISPF User's Guide* for more information about panel attributes.

Conversion Utility Syntax

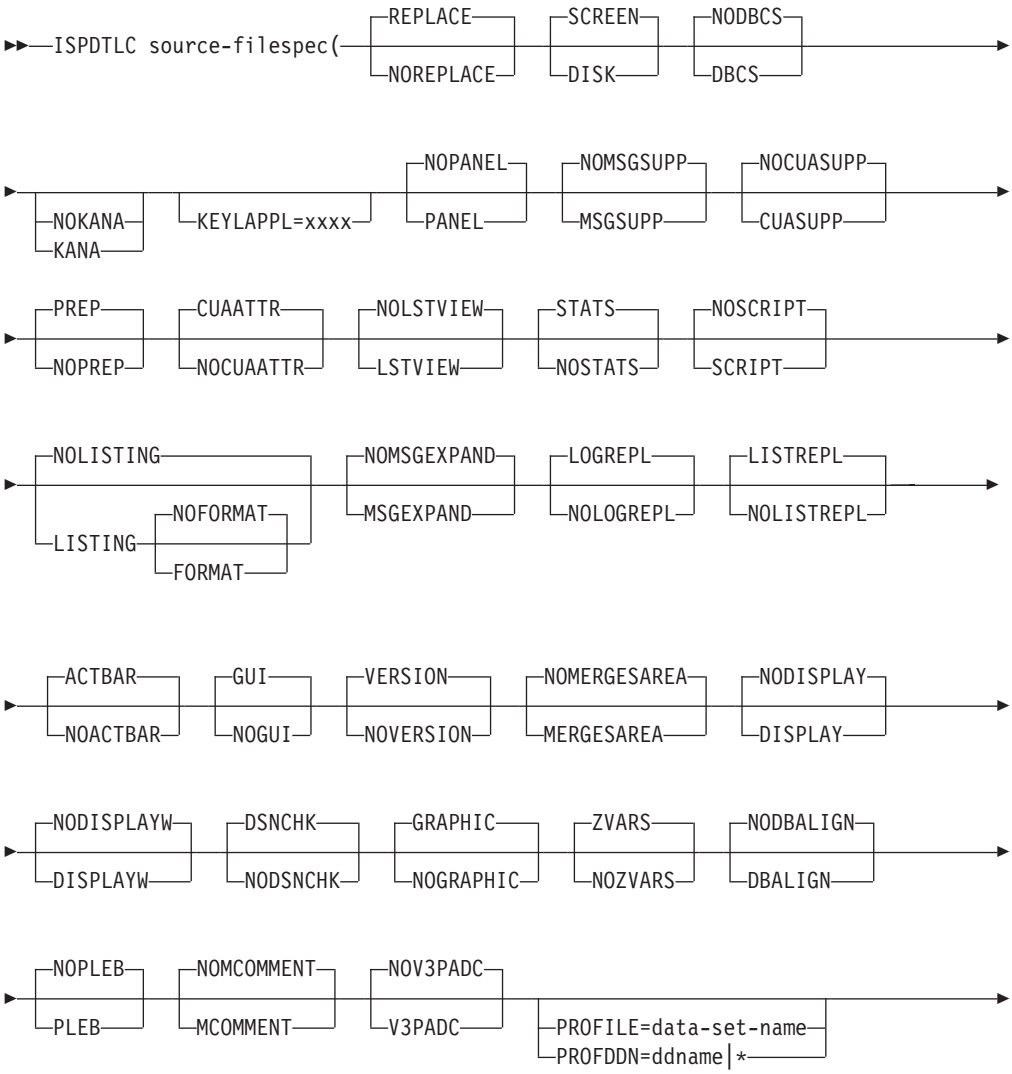
You can also invoke the conversion utility using the syntax discussed in this section. This feature provides compatibility with previous ISPDTLC releases and allows you to issue multiple calls from a user-specified EXEC file. To read the conversion utility syntax, see "Chapter 11. How to Read the Syntax Diagrams" on page 199 for more information.

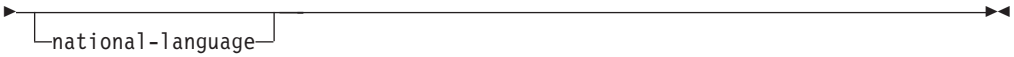
You can view the allowable syntax and a description of the options by entering the following command on the ISPF command line:

```
ISPDTLC ?
```

This command causes the general help panel to be displayed. The first line of information contains the ISPDTLC version, APAR, and PTF numbers.

This diagram shows the conversion utility syntax:





As denoted in the preceding diagram, when you specify options, a left parenthesis "(" is required before the first option. If you specify mutually exclusive options such as SCREEN and DISK, the conversion utility issues an error message and stops processing.

The syntax description follows:

source-filespec
Specify the *source-filespec* as a member of a partitioned data set (PDS) that contains the DTL source to be converted to ISPF dialog elements. The first-level qualifier is the "user ID" and the second-level qualifier is "GML" for the input data set name unless the PROFILE option is specified to override the default.

Note: The conversion utility output is stored as commands, keylists, messages and panels. A single source file might result in any or all of these objects. The source file might contain multiple command tables, keylists, message members or panels. The names for the output objects are provided by the CMDTBL, KEYL, MSGMBR, PANEL, and HELP tags. See the descriptions of these tags for additional information.

REPLACE | NOREPLACE
Indicates whether members generated by the conversion utility will replace existing members of the same name. If you specify NOREPLACE, the conversion utility issues a warning message for each existing member with the same name, but does not overwrite the existing member. If you specify REPLACE, the conversion utility overwrites any existing member with the same name. REPLACE and NOREPLACE affect keylists, commands, messages, panels, and SCRIPT files.

SCREEN | DISK
Indicates where to send information, warning, and error messages that occur while running the conversion utility. If you specify SCREEN (the default), conversion messages are sent to the display screen. If you specify DISK, conversion messages are sent to the designated log file.

Note: If your messages are not being written to the ISPF log, the specified conversion utility log file must be pre-allocated. If your messages are being written to the ISPF log, the ISPF Settings option must specify that an ISPF log is to be created.

Running the conversion utility with the DISK option causes additional messages to be appended to the existing sequential ISPDTLC log file or the ISPF log. When using the conversion utility log file, a separator record indicating the date and time of the execution is written to the log file before any messages.

Messages are written to the screen automatically when:
- The conversion utility detects errors during initialization.
- System I/O errors occur.

DBCS | NODBCS
Indicates whether or not DBCS validation is performed on tag text following the tag suffix ">". Errors found during DBCS validation cause the conversion

utility to issue error or warning messages. DBCS shift-out and shift-in characters are considered part of the text, thereby contributing to the length of the text.

Attention: DBCS strings cannot span records. That is, DBCS shift-out and shift-in characters (shift-in characters end the DBCS string) must be on the same record. The conversion utility ends with a severe error for incorrectly formed DBCS strings. If DBCS is specified and no language is specified, the default language is Japanese.

KANA | NOKANA
Indicates whether the KANA keyword will be added to the)BODY statement on panels and the message ID line of messages. There is no default. If KANA is specified and no language is specified, the default language is Japanese. Refer to the *ISPF User's Guide* for more information.

KEYLAPPL=xxxx
The KEYLAPPL=xxxx option, where "xxxx" is equal to the 1–4 character application ID, must be specified when the user includes a key list or lists in the DTL source and the APPLID attribute is omitted on the KEYL tag. The application ID is used by the conversion utility to write to the correct key list file.

Note: You cannot use "ISP" as an application ID, because the conversion utility is running as an ISP application.
Refer to *ISPF User's Guide* for restrictions on updating key lists.

PANEL | NOPANEL
The PANEL keyword forces the conversion utility to display the invocation panel even if a *source-filespec* has been entered. The PANEL keyword is disregarded when the conversion utility is running in a batch job.

MSGSUPP | NOMSGSUPP
The MSGSUPP keyword causes the conversion utility to suppress warning messages concerning panel formatting.

CUASUPP | NOCUASUPP
The CUASUPP keyword causes the conversion utility to suppress warning messages concerning CUA Architecture non-compliance.

PREP | NOPREP
The NOPREP keyword causes the preprocessing of the output panel to be bypassed. Panel output is stored in ISPF panel format.

CUAATTR | NOCUAATTR
The NOCUAATTR keyword forces the conversion utility to create panels with attribute definitions compatible with ISPF Version 3.1 and Version 3.2. CUAATTR causes panels to be created using CUA attribute types as defined in the *ISPF Dialog Developer's Guide and Reference*

Note: If you specify NOCUAATTR, the conversion utility will issue a message and change the default GRAPHIC option to NOGRAPHIC because GRAPHIC support is implemented only for CUA attributes.

LSTVIEW | NOLSTVIEW
The LSTVIEW keyword causes the conversion utility to display the "converting source file" message in line mode when the user has routed the log file messages to DISK. NOLSTVIEW causes the "converting source file" message to be displayed as a long message in full screen mode. The NOLSTVIEW

keyword is disregarded when the conversion utility is running in a batch job; the “converting source file” message is written to file SYSTSPRT.

STATS | NOSTATS
The NOSTATS keyword causes the conversion utility to bypass the creation of member statistics on created panels and messages. STATS and NOSTATS affect messages, panels, and SCRIPT files.

SCRIPT | NOSCRIPT
The SCRIPT keyword causes the conversion utility to create a panel image template as a member of a file allocated to DTLSCR. The panel image template has BookMaster tags included so that it may be incorporated into documentation files. Input and output fields in the panel image are shown as underscores. Run-time substitution variables are shown as “&varname”. Editing will be required to supply appropriate information for input and output fields and “&varname” values.

Note: The specified conversion utility SCRIPT output file must be preallocated.

LISTING | NOLISTING
The LISTING keyword causes the conversion utility to create a list file of the processed source GML records. This file is allocated to DTLLIST or if no file name is provided to the conversion utility, the list is added to the standard ISPF list data set. The file you provide can be in either sequential or partitioned format.

Note: If your messages are not being written to the ISPF list file, the specified conversion utility list file must be preallocated.

Indentation of nested tags (to a limit of 30 columns) is provided for readability. The listing is limited to an 80-column format. Tag contents that would extend beyond the right column are flowed to multiple lines.

The formatted listing is unchanged from the original DTL source file except for indentation processing.

FORMAT | NOFORMAT
The FORMAT keyword causes the conversion utility to create a list file of the source GML records after entity substitution is performed. (The FORMAT keyword implies the LISTING keyword.) The number at the left side of the list indicates the file nest level. If the LISTING keyword is specified in combination with the NOFORMAT keyword, all substitution is bypassed and the listing can be used as a formatted input GML file.

MSGEXPAND | NOMSGEXPAND
The MSGEXPAND keyword causes the conversion utility to expand the warning and error messages to include an indicator of the major type of tag in process (PANEL, HELP, KEYL, MSGMBR, CMDTBL) along with the object name.

LOGREPL | NOLOGREPL
Indicates whether members generated by the conversion utility will replace existing log file PDS members of the same name. If you specify NOLOGREPL, the conversion utility issues a warning message for each existing member with the same name but will not overwrite the existing member.

LISTREPL | NOLISTREPL
Indicates whether members generated by the conversion utility will replace existing list file PDS members of the same name. If you specify NOLISTREPL,

the conversion utility issues a warning message for each existing member with the same name but will not overwrite the existing member.

ACTBAR | NOACTBAR
Indicates whether the ISPF panel statements for action bars will be added to the generated panel. If you specify NOACTBAR, the panel sections for)ABC,)ABCINIT, and)ABCPROC and the action bar lines from the panel body are not added to the output panel. (The DTL source for action bar creation is syntax-checked in all cases.)

If a PANEL tag includes the keyword ACTBAR, this option is ignored for that panel.

GUI | NOGUI
The NOGUI keyword causes the GUI display mode panel keywords for mnemonics and check boxes to be removed from the generated panel.

If you specify MNEMGEN=YES on the AB tag or CHKBOX=YES on the SELFLD tag, this option is ignored for the specified tag. This option can be overridden by specifying the TYPE attribute on the PANEL tag.

VERSION | NOVERSION
Indicates whether the ISPDTLC version number, maintenance level, and member creation date and time are added as comments following the)END panel statement and the last message of a message member. In addition, VERSION causes the conversion language (ENGLISH, GERMAN, JAPANESE, and so on), Panel ID, and ISPF version to be added to the)ATTR panel statement line as a comment. If you specify NOVERSION, the comments are not added to the generated panel or message.

Note: If the PREP conversion option has been specified, the comments will not be part of the final panel because they will not be processed by the ISPPREP utility.

NOMERGESAREA | MERGESAREA
Indicates whether scrollable areas will be merged into panel body sections. Merge occurs only when the entire scrollable area can be contained within the panel body, allowing for the function key area. This option can be overridden by specifying the MERGESAREA attribute on the HELP or PANEL tag.

NODISPLAY | DISPLAY

Indicates whether the converted panel will be displayed by the conversion utility immediately after the panel is created. The display will be in full screen format. The DISPLAY keyword is disregarded when the conversion utility is running in a batch job.

Note: If you specify DISPLAY, ISPDTLC must be run in test mode (Option 7) to force display processing to use the current generated panel. An error message is issued if ISPDTLC is not being run in test mode and this option is specified.

DISPLAY causes each converted panel to be displayed until the user enters ***DISPLAY OFF*** on the command line of a displayed panel or selects option 2 from the display control panel. The control panel is displayed periodically, according to the interval specified in the **DISPLAY(W) option check interval** field on the invocation panel, or from the **Miscellaneous** choice on the **Options** action bar choice.

NODISPLAYW | DISPLAYW

Indicates whether the converted panel will be displayed by the conversion utility immediately after the panel is created. The display will be within a window. The DISPLAYW keyword is disregarded when the conversion utility is running in a batch job.

Note: If you specify DISPLAYW, ISPDTLC must be run in test mode (Option 7) to force display processing to use the current generated panel. An error message is issued if ISPDTLC is not being run in test mode and this option is specified.

DISPLAYW causes each converted panel to be displayed until the user enters ***DISPLAY OFF*** on the command line of a displayed panel or selects option 2 from the display control panel. The control panel is displayed periodically, according to the interval specified in the **DISPLAY(W) option check interval** field on the invocation panel, or from the **Miscellaneous** choice on the **Options** action bar choice.

DSNCHK | NODSNCHK
Indicates whether file validation will be performed on the files specified on the interactive panel after the first conversion cycle has been completed. If you specify NODSNCHK and any specified file is unavailable, the conversion will fail when the conversion utility attempts to use the file. The NODSNCHK keyword is disregarded when the conversion utility is running in a batch job.

GRAPHIC | NOGRAPHIC
Indicates, for host display only, whether the action bar separator line and visible horizontal divider lines will display as dashed lines or as solid lines. The GRAPHIC option can be overridden by the tag definition that generates the line. See tag attribute descriptions for

- "AB (Action Bar)" on page 211,
- "AREA (Area)" on page 223,
- "CHDIV (Choice Divider)" on page 242,
- "DA (Dynamic Area)" on page 287,
- "DIVIDER (Area Divider)" on page 296,
- "GA (Graphic Area)" on page 333,
- "GRPHDR (Group Header)" on page 337,
- "LSTFLD (List Field)" on page 382, and
- "LSTGRP (List Group)" on page 386

for information about the creation of action bar separator and various types of visible divider lines. (In GUI mode, the action bar separator always displays as a solid line, and divider lines always display as dashed lines.) If you specify NOGRAPHIC, the action bar separator line and visible divider lines will be created as dashed lines.

Note: If you specify NOCUAATTR, the conversion utility will issue a message and change the default GRAPHIC option to NOGRAPHIC because GRAPHIC support is implemented only for CUA attributes.

ZVARS | NOZVARS
Indicates whether variable names will be formatted as Z variables. If you specify NOZVARS, the variable name will be used in panel)BODY or)AREA formatting unless the variable name is longer than the defined field width.

DBALIGN | NODBALIGN
For DBCS language conversions only. Indicates whether fields with PMTLOC=ABOVE will be aligned so that the first position of the prompt text is formatted above the first position of the field.

PLEB | NOPLEB
Indicates whether leading blanks in ENTITY text strings are processed. This option is effective only for ENTITY definitions that do not specify the ***space*** keyword.

MCOMMENT | NOMCOMMENT
Indicates whether multiple line comment blocks, starting with **<! --** or **<: --** and ending with the first **-->** found are valid. Comment blocks can include DTL tags.

NOV3PADC | V3PADC
Indicates whether ISPDTLC Version 3 padding is added to global definitions for input fields in the)ATTR panel section. When ISPDTLC is invoked with the V3PADC option, the ISPF keyword PADC('_') is added to input attribute definitions if there is no PAD or PADC attribute specified on the PANEL tag.

PROFILE=data-set-name | PROFDDN=ddname | PROFDDN=*
The PROFILE or PROFDDN option provides access to the data set name that contains the conversion utility defined DD names and associated PDS/sequential file names to be used by the conversion utility during I/O. A sample profile member ISPDTLP is shipped in the ISPSLIB skeleton library.

The *data-set-name* value must be a fully qualified data set name that specifies either a sequential or a partitioned data set. If the profile entry is part of a partitioned dataset, then the member name must be included in the data-set-name specification.

The *ddname* value specifies a ddname allocated to a profile data set.

The "*" value specifies that the ddnames used in the conversion will be found as pre-allocated files. See page 195 for the ddnames used by ISPDTLC.

The profile data set and all data sets defined within the profile must be preallocated.

national-language
Specifies the language rules to be used for formatting the tag text. Supported language keywords are:

```
CHINESES     ENGLISH      ITALIAN      PORTUGUE     UPPERENG
CHINESET     FRENCH       JAPANESE     SGERMAN
DANISH       GERMAN       KOREAN       SPANISH
```

Note: ISPF must have been installed to support the language requested by the conversion utility.

Conversion Utility General Information

The output panel library can be defined as either fixed length or variable length. A fixed length record library must have a record length of 80, 132, or 160 bytes. Record lengths for variable length record libraries must be increased by 4. Variable length libraries defined with a record length other than 84, 136, or 164 are treated as the next smaller standard size. Thus, a variable length file of 255 bytes is treated as 164, and a variable length file of 100 bytes is treated as 84.

The NOPREP option directs the conversion utility to write the panels being processed directly to the specified panel output file in the ISPF source format. The overall width for the created panels is limited by the record length of the designated file. Thus, if you have specified a panel library with a fixed length of 80 bytes (or a variable length of 84 bytes), the maximum panel width allowed on the PANEL tag will be 80.

The PREP (default) option causes the creation of a temporary panel library to receive the ISPF source panel format file. The temporary library is created with a record length of 160 bytes. Multiple panels created in PREP mode are stored in the temporary library and converted through one call to ISPPREP. When all of the panels are converted, the temporary library is deleted. ISPPREP is called by the conversion utility when you do the following:

- Change the name of the output panel library on the ISPDTLC invocation panel and then convert another panel.
- Deselect the Preprocess Panel Output option on the ISPDTLC invocation panel and then convert another panel.
- Change the Generate Statistics on Panel/Message/Script Members option on the ISPDTLC invocation panel and then convert another panel.
- Enter "PREP" on the command line of the ISPDTLC invocation panel or select "PREP" from the 'Commands' action bar pull-down.
- Exit from the conversion utility.

ISPPREP is also called when:

- The number of extents of the temporary library exceeds 5.
- The number of members written to the temporary library exceeds 50.

ISPPREP output for panels longer than 80 bytes can be stored in a panel library with a fixed record length of 80 (or a variable record length of 84). Thus, you can create larger than standard panels in PREP mode while directing the final panel output to a library defined with a standard length. It is the developer's responsibility to ensure that the WIDTH specified on the PANEL tag is appropriate for the device intended to display the panel.

When the log or list files are specified as members of a partitioned data set, and the log or list file member name is specified as an asterisk (*) the member is written before the invocation panel is redisplayed. Otherwise, the log and/or list file is stored in memory (and added to for additional DTL source conversions) until one of the following occurs:

- The output log or list data set name is changed and another conversion is performed.
- The member name of the log or list file is changed on the invocation panel and another conversion is performed.
- The input DTL source member name is changed when the log or list member name is specified as an asterisk.
- You enter on the command line or select from the **Commands** action bar pull-down:
  SAVELOG
  to save the log file
  SAVELIST
  to save the list file
  SAVEALL
  to save both log and list files.
- You exit the conversion utility.

When the log file is specified as a partitioned data set, messages issued when the conversion utility ends are directed to the screen.

When the CANCEL command is entered, ISPDTLC displays a cancellation confirmation panel. This panel provides options for disposition of pending log and

list file members and for any panels to be processed by ISPPREP. An option is also provided to ignore the CANCEL command and resume ISPDTLC processing.

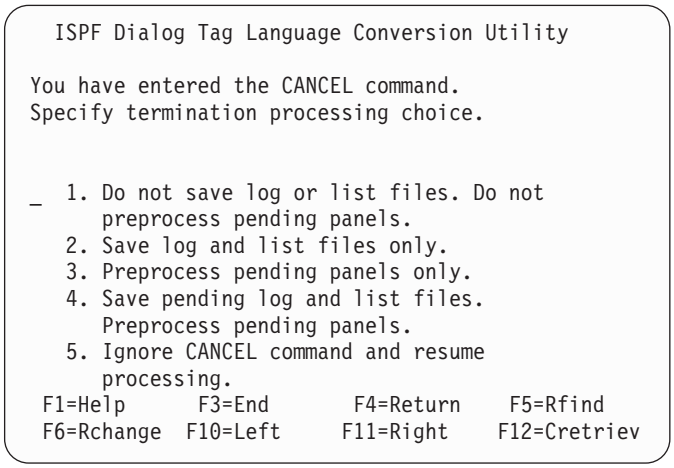


Figure 89. ISPF Dialog Tag Language Conversion Utility - Confirm Cancel

The panel appears with option 1 preselected. You may choose another option to save log and list files only, preprocess pending panels only, save log and list files and preprocess pending panels, or resume processing.

When you enter the SUBMIT command, ISPDTLC creates and submits a batch job, using the file names and options specified on the interactive panel. After the job is submitted, the interactive panel is redisplayed. The batch JCL file is built using the ISPF skeleton ISPDTLB.

You can also run ISPDTLC from ISPF options 4 and 5 and from the workplace member list.

Note: From the workplace member list, enter “T” (TSO) in front of the member name to be processed. On the TSO pop-up panel enter

```
"ISPDTLC / (PANEL RETURN" to run a foreground conversion or
"ISPDTLC / (PANEL SUBMIT" to submit a batch job.
```

After you complete the required ISPDTLC invocation panel fields and press Enter, the conversion runs or the job is submitted, and control is returned to the previous option.

Extremely large DTL input source files (source files that contain multiple panel, message, key list, and application command table definitions) might cause memory capacity to be exceeded. Should this occur, split the DTL input source file into multiple files with fewer panels, message members, key lists, or command table definitions or reduce the record length of the input source file.

When ISPDTLC is invoked recursively, that is, more than 1 time from the same ISPF screen, the following panel is displayed.

```
   ISPF Dialog Tag Language Conversion Utility

CAUTION:
ISPDTLC has been invoked recursively.
Results are not predictable.

Enter processing option.
_  1. CANCEL this invocation of ISPDTLC.
   2. Proceed with recursive execution.









 F1=Help     F3=End      F4=Return   F5=Rfind
 F6=Rchange F10=Left    F11=Right   F12=Cancel
```

Figure 90. ISPF Dialog Tag Language Conversion Utility - Recursive invoke

The panel appears with option 1 preselected. If you select option 2, the new invocation will be processed. Because of possible region size limitations, results are not predicable.

The recursive invocation check is based on the setting of a profile variable that is unique for each active screen. If the recursive check panel appears following an abend, the profile variable was not properly reset when the abend occurred. In this case, select option 2 to allow ISPDTLC to continue.

If the conversion utility is called without a *source-filespec* or if the PANEL option has been specified, the invocation panel is displayed. If other options have been specified, they are merged with the options from the profile before the display. The PROFILE option is disregarded when the invocation panel is displayed.

The *source-filespec* can be a special file which is a list of other files to be converted. When you use this option, you can convert multiple panels with a single call to the conversion utility. The format of the file list is :

```
DTLLST source-filespec 1
DTLLST source-filespec 2
:
```

The format of *source-filespec* is the same as any other call to the conversion utility. Duplicate *source-filespec* names within DTLLST are ignored.

The *national-language* selection UPPERENG causes the conversion utility to use the uppercase version of the ENGLISH program literals. In addition, the tag text for all tags except <SOURCE> is translated to uppercase during the conversion process.

The *national-language* selection SGERMAN causes the conversion utility to use a special German-to-Swiss German conversion routine to create Swiss German panels from either German or Swiss German DTL source files.

ISPF Conversion Utility Messages

During processing, the conversion utility can issue information, warning, and error messages. For unsupported DTL tags and attributes that generate warning messages, the conversion utility either ignores the tag or attribute, or sets attribute values to the conversion utility defaults. If the conversion causes error messages, the conversion utility will not generate the ISPF file (key list, panel, application command table, or message member) that would have been created had the error not occurred.

In the message listing, the line numbers displayed in the messages might not always match the line numbers of the source file that caused the message. This occurs because the conversion utility must sometimes continue to read the source file until it encounters an end tag or a new tag before issuing a message. You should be able to determine which source line created the message by examining the DTL source file.

There are two options required to suppress all noncritical messages.

- The MSGSUPP option is used to suppress messages related to ISPDTLC formatting.
- The CUASUPP option is used to suppress messages related to CUA architecture deviations allowed by ISPDTLC. Examples include nonstandard use of F1/F13, F3/F15, and F12/F24 keylist commands, and the use of the SMSG attribute on the MSG tag to create a short message.

When each DTL source file conversion is completed, the conversion utility issues a message listing the number of warning and error messages generated. If the MSGSUPP or CUASUPP option(s) have been specified, an additional message is issued with the total number of messages suppressed.

When the conversion utility is finished, it issues a message listing the total number of warning and error messages generated. If the MSGSUPP or CUASUPP option(s) have been specified, a message is issued with the total number of messages suppressed. The end of job messages listing the total number of messages are placed in the ISPF log file, if the log file is available; otherwise the overall totals are written to the terminal.

Return Codes

The following list of return codes explains the results of the conversion invocation.

0 No warnings, errors, or severe errors

1 All messages were suppressed.

4 CANCEL command ended ISPDTLC

8 Only warnings were found

16 At least one DTL conversion had at least one error

20 At least one DTL conversion ended with a severe error.

For multiple conversions, the highest return code is used.

Conversion Results

The results of the conversion are placed in the shared pool.

- The variable ZDTLRC contains the return code.

- The variable ZDTLNWRN contains the number of warning messages.
- The variable ZDTLNERR contains the number of error messages.
- The variable ZDTLNSUP contains the number of suppressed messages.

Conversion Utility File Names

The conversion utility is shipped as a REXX exec on the ISPF product tape.

The ISPDTLC exec can reside in a CLIST data set allocated to SYSPROC or in an EXEC data set allocated to SYSEXEC. For more information on the use of REXX execs on MVS, refer to the *TSO/E Version 2 REXX User's Guide*.

Additional Requirements:

- All data sets must be allocated prior to running the conversion utility. In addition, the conversion utility uses ISPF services to produce command table and key list output, which means that a partitioned data set must be allocated to ISPTABL. Refer to the section on allocating ISPF libraries in the *ISPF User's Guide* for more information.
- To allow the user to specify the source and destination data sets when using the conversion utility syntax, seven DD names have been reserved in an allocation profile with associated data set names to be provided by the user.

  Note: ISPDTLC profiles from previous releases can be used without change. However, a warning message is issued if the DTLMIN or DTLNLS DD name records are encountered.
- A sample profile member ISPDTLP is shipped in the ISPSLIB skeleton library. You can modify the data set names for installation or user use. DTL format comments (**<!--comment text-->** or (**<:--comment text-->**) can be used in the profile data set or member. Do not modify the DDNAMEs in the following table (column one). A sample user updated profile member follows:

DDNAME	Data Set
DTLGML	any.GML.input
DTLPAN	your.panel.output
DTLMSG	your.msg.output
DTLLOG [6]	your.log.output
DTLLIST [6]	your.list.output
DTLSCR	your.script.output
DTLTAB	your.table.output

  DTLGML is the input file to the conversion utility. The last 6 files are for output and are usually the user's own data sets.
- For compatibility with previous ISPDTLC releases, the user can provide the allocation profile name on invocation:

  ```
  ISPDTLC source-filespec (disk PROFILE=User.profile
  ```

  The data set name following the PROFILE keyword must be a fully-qualified data set name. When specifying the data set name, do not include quotes.

6. The sequential data set name associated with the DTLLOG and DTLLIST ddnames should have the same characteristics and attributes as the LOG and LIST data sets for ISPF.

The profile data-set-name can specify either a sequential or a partitioned data set. If the profile entry is part of a partitioned dataset, then the member name must be included in the data-set-name specification. The profile data set and all data sets defined within the profile must be pre-allocated.

An example of the default data set names used for the conversion utility is shown in the following table. USERID is the user's TSO prefix.

DDNAME	Data Set	Type	Description
DTLGML	userid.GML	PDS	The DTL source PDS where GML members reside.
DTLPAN	userid.PANELS or NULLFILE or DUMMY	PDS	PDS for panel member output. May be specified as NULLFILE or DUMMY for cases where no panel output is required.
DTLMSG	userid.MSGS or NULLFILE or DUMMY	PDS	PDS for message member output. May be specified as NULLFILE or DUMMY for cases where no message output is required.
DTLLOG	userid.ISPDTLC.LOG userid.LOGLIB(logmem)	SEQ or PDS	Optional. User's log data set for conversion utility messages. If not specified, log messages are written to the standard ISPF log data set. If file is a PDS, member name must be included in the data set name specification.
DTLLIST	userid.ISPDTLC.LIST userid.LISTLIB(listmem)	SEQ or PDS	Optional. User's list data set for conversion utility messages. If not specified, list messages are written to the standard ISPF list data set. If file is a PDS, member name must be included in the data set name specification.
DTLSCR	userid.SCRIPT	PDS	Optional. PDS for panel member documentation output. The DTLSCR data set is required only if the SCRIPT option is specified.
DTLTAB	userid.TABLES	PDS	Optional. PDS for keylist and command table output. If specified, a LIBDEF is performed for ISPTLIB and ISPTABL and the keylist and command table output is written to the data set.

The profile can contain multiple entries for each DD name. For output files, the first valid data set name in the profile is used. For the input GML file, each data set is checked in the order they are found in the profile for the member name specified. The first match by member name is used as the file to be converted.

When the data set associated with either the DTLLOG or DTLLIST ddname in the profile is a PDS, the member name may be a single asterisk. When the asterisk notation is present, the conversion utility uses the same name for the log or list file as the source GML member name.

Part 2. Dialog Tag Language (DTL) Reference

This section contains the following chapters:

Chapter 11. How to Read the Syntax Diagrams
This chapter describes the elements that make up a syntax diagram.

Chapter 12. Markup Declarations and DTL Macro Reference
This chapter contains a reference listing for each DTL markup declaration.

Chapter 13. Tag Reference
This chapter contains a reference listing for each DTL tag.

Each reference listing in this section contains a syntax diagram and attribute definition list, as well as a description and examples of usage.

© Copyright IBM Corp. 1989, 2001

Chapter 11. How to Read the Syntax Diagrams

Throughout this book, syntax is described using the structure defined below.

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

  Symbol
  Description

  ►►— indicates the beginning of a statement.

  —► indicates that the statement syntax is continued on the next line.

  ►— indicates that a statement is continued from the previous line.

  —►◄ indicates the end of a statement.

  Diagrams of syntactical units other than complete statements start with the ►— symbol and end with the —► symbol.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.

- Items positioned above the syntax diagram line are default parameters.

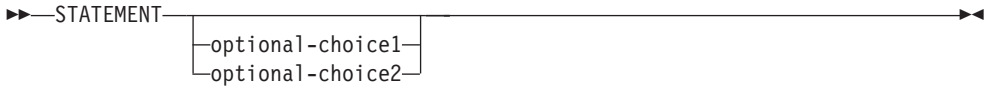
- If you can choose from two or more items, these items appear vertically, in a stack.

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

  If a default is also available, it appears above the main line.

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

© Copyright IBM Corp. 1989, 2001



- An arrow returning to the left above the item indicates an item that you can repeat. Required items appear on the main line and optional items appear below the main line.

  A repeat arrow indicates that you can make more than one choice from the grouped items, or repeat a single item.
- Keywords appear in uppercase (for example, `PARM1`). However, they can be uppercase or lowercase when they are entered. They must be spelled exactly as shown. Variables and acceptable values appear in all lowercase letters (for example, `parmx`). They represent names or values that you supply. Keywords and keywords followed by parameters (for example, `MSG=message-id`) can be coded in any order.
- If punctuation marks, parentheses, arithmetic operators, or other symbols are shown, you must enter them as part of the syntax.

Chapter 12. Markup Declarations and DTL Macro Reference

This chapter provides you with a detailed look at the following topics:
- Document-type declaration
- Entity declarations.
- Sample Entity definitions.
- DTL Macros.

Document-Type Declaration

The document-type declaration (DOCTYPE) identifies the source file document type and the rules the source file must follow.

DOCTYPE
Indicates that this is a document-type declaration.

DM
Indicates that this is a DTL source file defining dialog elements.

SYSTEM
Indicates that the rules for the DOCTYPE are contained in an external file.

[| (
Indicates the beginning of the declaration subset. Either the left bracket or the open parenthesis can be used to begin the declaration subset. The declaration subset can contain entity declarations and parameter entity references. If the left bracket is coded, it must be the hex value 'AD'.

entity-declarations
The entity declarations you define for the source file must be coded within the declaration subset. "Entity Declarations" on page 202 contains a complete description of entity declarations.

] |)
Indicates the end of the declaration subset. Either the right bracket or the close parenthesis can be used to end the declaration subset. If the right bracket is coded, it must be the hex value 'BD'.

Description

A document-type declaration identifies the source file document type and rules the source file must follow.

The DOCTYPE declaration must appear in a DTL source file before any tag markup, although it can be preceded by comments. Files that are embedded in a source file intended for compilation cannot contain a DOCTYPE declaration.

Example

The DOCTYPE statement declares the source file as a DM type file.

```
<!doctype dm system>

<varclass name=varc type='char 10'>

<varlist>
  <vardcl name=vard varclass=varc>
</varlist>
```

© Copyright IBM Corp. 1989, 2001

```
<panel name=panel>
⋮
```

Entity Declarations

Entities are symbolic names that are used to insert text into a file.

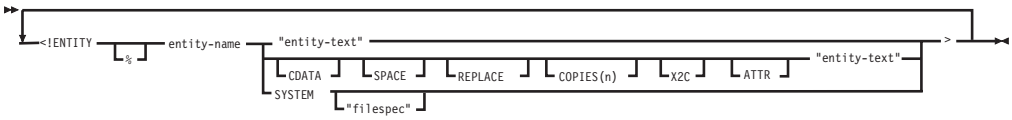


Figure 91. Entity Declarations

ENTITY
Indicates this is an entity declaration.

% Indicates a parameter entity declaration, which must be followed by at least one space.

entity-name
The name of the entity. It must follow these rules:

- Length
  - 1–8 for file imbed entity names
  - 1–8 for parameter entity names
  - 1–17 for other entity names
- The first character must be A–Z, a–z, @, #, or $.
- Remaining characters, if any, can be A–Z, a–z, @, #, $, or 0–9.

  When an entity name is more than 8 bytes in length, one or more of the remaining characters must be an underscore.
- Entity names are case-sensitive.
- The entity name for a parameter entity can be specified as a variable name (that is, **%&varname;**). The resolved name must follow the parameter entity naming rules.

CDATA
Indicates that any delimiter characters in *entity-text* will not be interpreted as delimiters. This allows you to define entities with tags in *entity-text* that will not be interpreted as tags.

For example the entity-text "<panel>" is not interpreted as the PANEL tag if the CDATA keyword is used.

The effect of CDATA is to delay substitution of the variable until all other text manipulation is completed. For example, you should use CDATA to specify an *entity-text* string of blanks as normal text processing removes leading and trailing blanks from text strings.

Note: CDATA cannot be used with parameter entities.

SPACE
Indicates that *entity-text* which spans multiple DTL source file records will be formatted like the <p> tag. (Leading and trailing blanks on *entity-text* lines will be compressed to a single blank character.) In addition, multiple blanks between words of *entity-text* will be compressed to a single blank character.

REPLACE
Indicates that the current *entity-text* is to replace any previous definition of the same entity name.

COPIES(n)
Indicates that the *entity-text* is to be expanded by repeating the provided text *n* times. For example, including COPIES(5) as a keyword with the text specified as "*" causes the entity text to be processed as "*****".

X2C
Indicates that the specified hex format *entity-text* is to be converted to character format. Non-valid hex values are processed as a regular entity character string.

ATTR
Indicates that the specified *entity-text* is a CUA text attribute. Valid values are: CH, CT, DT, ET, FP, NT, PIN, PT, SAC, SI, WASL, and WT. These values are converted to their corresponding attribute byte. Non-valid attribute codes are processed as a regular entity character string.

"entity-text"
The text associated with the entity reference. The text must be enclosed in single or double quotes. The length of the *entity-text* must be less than or equal to 253 bytes.

SYSTEM
Indicates this entity refers to an external file.

"filespec"
The name of the file the entity refers to. The name must be enclosed in single or double quotes. If this is not specified, it defaults to the name of the entity.

The SYSTEM parameter can optionally be followed by the filename for the included file. The file name for MVS is a member name for a file provided on the invocation panel or specified as "DTLGML" entries in the ISPDTLC profile.

Description

Entities are symbolic names that are used to insert text into a file. The text that an entity refers to can be a simple string of characters or it can be the text from an entire file.

An entity reference is used to insert the text associated with the entity. Entities must be declared in the declaration subset of the DOCTYPE declaration before they can be referred to. To refer to the entity in the source file, the entity name is preceded by an ampersand (&) to indicate it is an entity or percent (%) to indicate it is a parameter entity. Both types of entities are ended with a semicolon (;). A blank or the end of the line can be used to end the entity reference instead of the semicolon.

Because entity declarations can only be made within the declaration subset, the parameter entity is the only way to embed a file of entity declarations. Parameter entities are used when an entity reference is needed in the declaration subset. References to parameter entities can only be made in the declaration subset.

References to entities can be made anywhere in the source file *after* the end of the DOCTYPE declaration.

Note: To refer to an entity within a <SOURCE> tag in the source file, the entity name is preceded by a percent (%) instead of an ampersand (&).

Because entity names are case-sensitive, ensure that references to entities are specified correctly.

Conditions

Entities that are declared do not have to be referred to.

Example

This example uses both entities and parameter entities. It embeds the file GLBENT with global entity declarations, and a file with tags and text. It also uses entities and parameter entities that refer to text strings.

The first entity declaration declares the “glbent” parameter entity as an external file.

The file name is defaulted to GLBENT. A parameter entity is used because this file contains entity declarations. Because entity declarations can only be made in the declaration subset, the GLBENT file is embedded with an entity reference within the declaration subset. The entity declarations in GLBENT are for text that is used at the top and bottom of the panel. The “header” entity declaration refers to an external file, and the “footer” is a text string. Both of these entities are referred to in the source file.

The second entity declaration, for “list”, is also a parameter entity. This declaration refers to a string, not an external file. The text is the SL tag name, which is referred to in the next two entity declarations. These two declarations, “slist” and “elist”, are used as the SL start and end tags. They are defined as entities so the type of list can be changed in one place. To change the list type from a simple list (SL) to an unordered list (UL), change the parameter entity “list” from SL to UL.

This is the content of the source file:

```
<!DOCTYPE DM SYSTEM [
<!ENTITY % glbent SYSTEM -- declaration of global entity file -->
%glbent;<!-- Embeds the global entity file -->
<!ENTITY % list "sl" -- type of list -->
<!ENTITY slist "<%list;>" -- type of list start tag. -- >
<!ENTITY elist "</%list;>" -- type of list end tag. -- >
]>

<panel name=showlist depth=22 width=45>Show Departments
  <area>
    <info width=40>
      &header;
      <p>The floors and departments are shown below:
      &slist;
        <li>First floor
          &slist;
            <li>Toys
            <li>Electronics
          &elist;
        <li>Second floor
          &slist;
            <li>Boys clothes
            <li>Girls clothes
          &elist;
      &elist;
      &footer;
    </info>
  </area>
</panel>
```

This is the content of the embedded file GLBENT:

```
<!ENTITY header SYSTEM "coname">
<!ENTITY footer "<p> We're always glad to help!">
```

This is the content of the embedded file CONAME:

```
<lines>
Jeff's Children's World
Barnett, NC
</lines>
```

Figure 92 shows the formatted result:

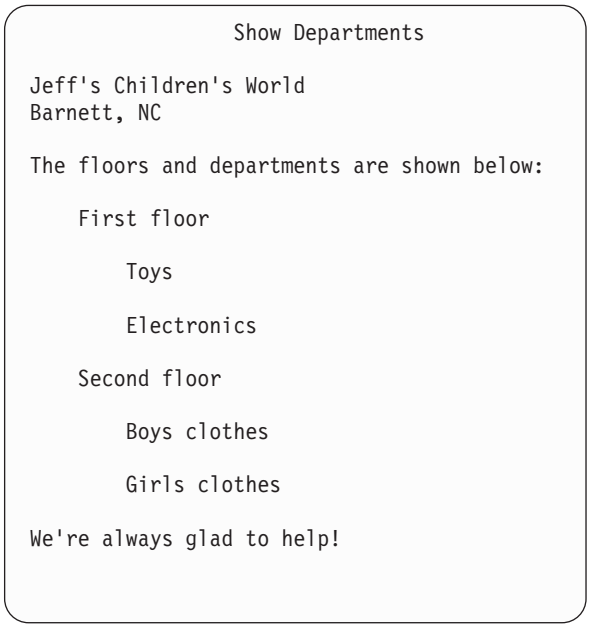


Figure 92. Entities and Parameter Entities

Sample Entity Definitions

The tag examples in "Chapter 13. Tag Reference" on page 211 use entity definitions to create the sample panels. The entities used are called SAMPABC (to define the action bar); SAMPVAR1, SAMPVAR2, and SAMPVAR3 (to provide VARCLASS and VARLIST definitions); and and SAMPBODY (to provide a panel body section).

The DTL definitions follow:

SAMPABC:

```
<ABC>File
  <PDC>Add Entry
    <ACTION RUN=add>
  <PDC>Delete Entry
    <ACTION RUN=delete>
  <PDC>Update Entry
    <ACTION RUN=update>
  <PDC>Exit
    <ACTION RUN=exit>
<ABC>Search
  <PDC CHECKVAR=whchsrch MATCH=1>Search on name
    <ACTION SETVAR=whchsrch VALUE=1>
    <ACTION RUN=search>
```

```
  <PDC CHECKVAR=whchsrch MATCH=2>Search on card number
    <ACTION SETVAR=whchsrch VALUE=2>
    <ACTION RUN=search>
<ABC>Help
  <PDC>Extended Help...
    <ACTION RUN=exhelp>
  <PDC>Keys Help...
    <ACTION RUN=keyshelp>
```

SAMPVAR1:

```
<VARCLASS NAME=date    TYPE='char 8'>
<VARCLASS NAME=numcls  TYPE='numeric 7'>
<VARCLASS NAME=namecls TYPE='char 25'>
<VARCLASS NAME=char1cls TYPE='char 1'>
<VARCLASS NAME=char7cls TYPE='char 7'>

<VARLIST>
  <VARDCL NAME=whchsrch VARCLASS=char1cls>
  <VARDCL NAME=curdate VARCLASS=date>
  <VARDCL NAME=cardno  VARCLASS=numcls>
  <VARDCL NAME=name    VARCLASS=namecls>
  <VARDCL NAME=address VARCLASS=namecls>
  <VARDCL NAME=cardsel VARCLASS=char1cls>
  <VARDCL NAME=card  VARCLASS=char7cls>
  <VARDCL NAME=north VARCLASS=char1cls>
  <VARDCL NAME=south VARCLASS=char1cls>
  <VARDCL NAME=east  VARCLASS=char1cls>
  <VARDCL NAME=west  VARCLASS=char1cls>
  <VARDCL NAME=nth VARCLASS=char1cls>
  <VARDCL NAME=sth VARCLASS=char1cls>
  <VARDCL NAME=est VARCLASS=char1cls>
  <VARDCL NAME=wst VARCLASS=char1cls>
</VARLIST>
```

SAMPVAR2:

```
<VARCLASS NAME=casecls TYPE='char 7'>
<VARCLASS NAME=namecls TYPE='char 25'>
<VARCLASS NAME=addrcls TYPE='char 25'>
<VARCLASS NAME=char1cls TYPE='char 1'>
<VARCLASS NAME=char2cls TYPE='char 2'>

<VARLIST>
  <VARDCL NAME=caseno  VARCLASS=casecls>
  <VARDCL NAME=name    VARCLASS=namecls>
  <VARDCL NAME=address VARCLASS=addrcls>
  <VARDCL NAME=casesel VARCLASS=char2cls>
  <VARDCL NAME=patin   VARCLASS=char1cls>
  <VARDCL NAME=defa    VARCLASS=char1cls>
  <VARDCL NAME=cont    VARCLASS=char1cls>
  <VARDCL NAME=priv    VARCLASS=char1cls>
  <VARDCL NAME=incr    VARCLASS=char1cls>
  <VARDCL NAME=disp    VARCLASS=char1cls>
  <VARDCL NAME=fraud   VARCLASS=char1cls>
</VARLIST>
```

SAMPVAR3:

```
<VARCLASS NAME=namecls  TYPE='char 7'>
<VARCLASS NAME=char1cls TYPE='char 1'>
<VARCLASS NAME=char2cls TYPE='char 2'>

<VARLIST>
  <VARDCL NAME=file  VARCLASS=namecls>
  <VARDCL NAME=type  VARCLASS=char2cls>
  <VARDCL NAME=marg  VARCLASS=char2cls>
  <VARDCL NAME=copy  VARCLASS=char2cls>
  <VARDCL NAME=duplx VARCLASS=char1cls>
</VARLIST>
```

SAMPBODY:

```
<TOPINST>Type in patron's name and card number (if applicable).
<TOPINST>Then select an action bar choice.
<AREA>
  <DTAFLD DATAVAR=curdate PMTWIDTH=12 ENTWIDTH=8 USAGE=out>Date
  <DTAFLD DATAVAR=cardno PMTWIDTH=12 ENTWIDTH=7 DESWIDTH=25>Card No
    <DTAFLDD>(A 7-digit number)
  <DTAFLD DATAVAR=name PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25>Name
    <DTAFLDD>(Last, First, M.I.)
  <DTAFLD DATAVAR=address PMTWIDTH=12 ENTWIDTH=25>Address
  <DIVIDER>
  <REGION DIR=horiz>
  <SELFLD NAME=cardsel PMTWIDTH=30 SELWIDTH=38>Choose
  one of the following
    <CHOICE CHECKVAR=CARD MATCH=NEW>New
    <CHOICE CHECKVAR=CARD MATCH=RENEW>Renewal
    <CHOICE CHECKVAR=CARD MATCH=REPLACE>Replacement
  </SELFLD>
  <SELFLD TYPE=multi PMTWIDTH=30 SELWIDTH=25>Check valid branches
    <CHOICE NAME=NORTH HELP=NTHHLP CHECKVAR=NTH>North Branch
    <CHOICE NAME=SOUTH HELP=STHHLP CHECKVAR=STH>South Branch
    <CHOICE NAME=EAST HELP=ESTHLP CHECKVAR=EST>East Branch
    <CHOICE NAME=WEST HELP=WSTHLP CHECKVAR=WST>West Branch
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>Enter a command
```

DTL Macros

A DTL macro is a DTL source member found in the concatenated DTL source libraries allocated as input to ISPDTLC. The macro member can be empty or it can contain any DTL tag coding, including DTL comments.

The macro member is imbedded into the current DTL source file when the macro name is encountered during conversion. The file imbed process is similar to Entity file imbed.

DTL macro tag syntax is similar to regular DTL tag syntax. You invoke a macro by specifying the macro member name using a special DTL tag open delimiter, like this:

```
<?macmemb>
```

The macro member name must conform to the DTL standard member name rules.

When ISPDTLC finds the **<?** open delimiter, a file imbed is performed on the specified member name. The reserved member name **dummy** can be specified to create a *no operation* (NOP) situation during the imbed cycle. If the specified member has no records, conversion continues with the next DTL source record.

The content of the macro member can be any valid DTL source input. The member can contain multiple tags and comment records just like any other DTL source file. DTL source file variables, sometimes referred to as entities, are substituted using standard entity processing.

An advantage to using the macro syntax instead of an entity file imbed is that you need not code the entity declaration for the file to be imbedded. ISPDTLC resolves the required information for you.

Another advantage is that you can specify entity values as part of the macro coding syntax, and bypass the coding of other entity delcarations. For example, if the entity variables ***subst_var_1***, ***subst_var_2***, and ***subst_var_3*** were coded within the macro using standard DTL syntax (that is, &subst_var_1;,&subst_var_2;, and &subst_var_3;), you could invoke the macro and specify the substitution values like this:

```
<?macmemb subst_var_1=subvalue1 subst_var_2=subvalue2
          subst_var_3=subvalue3>
```

ISPDTLC automatically defines the entities with the specified values. The values are stored using entity REPLACE processing, so that if a previous definition exists, it is overwrittten. The new definition remains in effect until replaced, and can be referenced by any other part of the DTL source file.

Macro tags placed within the document declaration function use the same rules as macro tags found after the document declaration. For example, you can use the macro syntax in place of parameter entities. The parameter entity (really a file of other entity definitions) member ***pentmem*** can be imbedded easily by coding

```
<?pentmem>
```

within the document declaration. This syntax replaces the more complicated parameter entity coding of

```
<:ENTITY % pentmem; system>
         %pentmem;
```

In another example, the macro syntax can be used in place of entity tags.

```
<?dummy   panel_title='ISPF macro example'
          panel_width=60
          panel_depth=18>
```

This syntax replaces multiple entity definitions:

```
<:ENTITY panel_title 'ISPF macro example'>
<:ENTITY panel_width '60'>
<:ENTITY panel_depth '18'>
```

In the previous example the macro name ***dummy*** is used to bypass the file imbed and enable the attribute resolution process to establish the entity values.

The macro name ***dummy*** can also be used within a macro definition to provide default values for macro entity variables. Example:

```
<:-- macro/include ISPZ@EX1 to format a 2 column example -->
<?dummy ?col1_indent=0 ?col1_width=30>

<region dir=horiz>
  region dir=vert width=&col1_width; indent=&col1_indent;>
    <pnlinst>&col1_text;
  </region>

  <region dir=vert>
    <pnlinst>&col2_text;
  </region>
</region>
```

In this example the entity variables ***col1_width*** and ***col1_indent*** have default values specified by the ***dummy*** tag. The special syntax '?variable=value' is used to provide the default values.

If you invoke the ISPZ@EX1 macro like this:

```
<?ispz@ex1 col1_text='text left' col2_text='text right'>
```

the default values for col1_width and col1_indent will be used.

If you invoke ISPZ@EX1 macro like this:

```
<?ispz@ex1 col1_width=40 col1_indent=4
    col1_text='text left' col2_text='text right'
```

the default values for col1_width and col1_indent will be overridden by those specified.

Entities

Chapter 13. Tag Reference

This chapter contains an alphabetical reference of the Dialog Tag Language (DTL) tags.

Each reference listing contains:
- A diagram of the valid syntax for the tag
- A list describing the tag attributes
- A description of the tag
- Conditions of usage
- A table of the tags that can be nested within the tag
- An example of how the tag is used within DTL source markup.

Rules for Variable Names

Variable names supplied as attribute values on DTL tags must have the following characteristics:
- 1–8 characters in length
- The first character must be A–Z, a–z, @, #, or $.
- Remaining characters, if any, can be A–Z, a–z, @, #, $, or 0–9.

Lowercase characters are translated to their uppercase equivalents

Names composed of valid characters that are longer than 8 bytes are truncated to 8 bytes. Names that are not valid are set to blank.

Rules for "%variable" Names

When a "%varname" notation is found as an attribute value, the "%varname" entry must have the following characteristics:
- 2–9 characters in length
- The first character is a "%".
- The second character must be A–Z, a–z, @, #, or $.
- Remaining characters, if any, can be A–Z, a–z, @, #, $, or 0–9.

Lowercase characters are translated to their uppercase equivalents

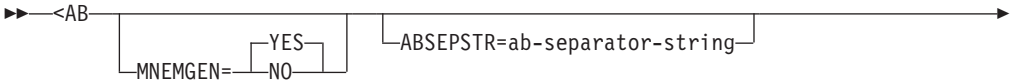
The first position of a valid name is replaced by an "&".

Names composed of valid characters that are longer than 9 bytes are truncated to 9 bytes. Names that are not valid are set to blank.

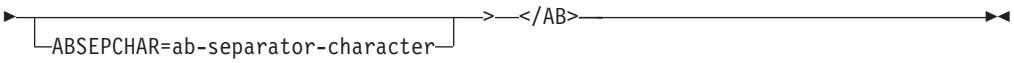
It is the responsibility of the application to provide a valid value in the variable before the panel is displayed.

AB (Action Bar)

The AB tag defines an action bar on an application panel.



© Copyright IBM Corp. 1989, 2001



MNEMGEN=YES | NO

Note: When the conversion utility is operating in DBCS mode, the default value for MNEMGEN is NO.

This attribute controls the automatic generation of mnemonic characters for the entire action bar. When MNEMGEN=NO, mnemonic characters are determined only by the use of the M tag within action bar or pull-down choice description text. See "Mnemonic Choice Selection" on page 40 and "M (Mnemonic)" on page 392 for additional information.

When MNEMGEN=YES, the NOGUI invocation option is ignored and mnemonics will be generated automatically.

ABSEPSTR=ab-separator-string
This attribute provides a string of data to be overlayed at the right end of the action bar separator line.

Note: This attribute is NOT recommended for general use because the action bar separator line is not displayed when operating in GUI mode.

ABSEPCHAR=ab-separator-character
This attribute provides a replacement character for the action bar separator line. When the GRAPHIC invocation option has been specified, the action bar separator will default to a solid line for host display. You can use the ABSEPCHAR attribute to provide a different character such as a dash.

Description

The AB tag defines an action bar on an application panel. The action bar appears on the panel above the panel title line. The action bar provides a way for users to view all actions that apply to the panel it is coded within.

The conversion utility inserts a line between the action bar and the panel title line. The GRAPHIC invocation option creates a solid line. NOGRAPHIC creates a dashed line. If required by the length or number of action bar choices, the conversion utility will format multiple lines for the action bar.

ABC tags, which you code within an AB definition, define application panel choices for the action bar. PDC tags, which you code within ABC tag definitions, define the action bar pull-down choices.

To define an action bar and its associated pull-downs, you code the AB tag (and other tags that define the action bar choices and pull-downs) within a PANEL definition.

Conditions

- The AB tag requires an end tag.
- You must code the AB tag within a PANEL definition. Each application panel can include only one action bar. See "PANEL (Panel)" on page 415 for a complete description of this tag.
- You must code at least one ABC tag within an action bar definition.
- To conform to CUA rules, you must include a help action bar choice.

Nested Tags

You can code the following tag within an AB definition:

Tag	Name	Usage	Page	Required
ABC	Action bar choice	Multiple	214	Yes

Example

The following markup contains the action bar markup for the application panel illustrated in Figure 93 on page 214.

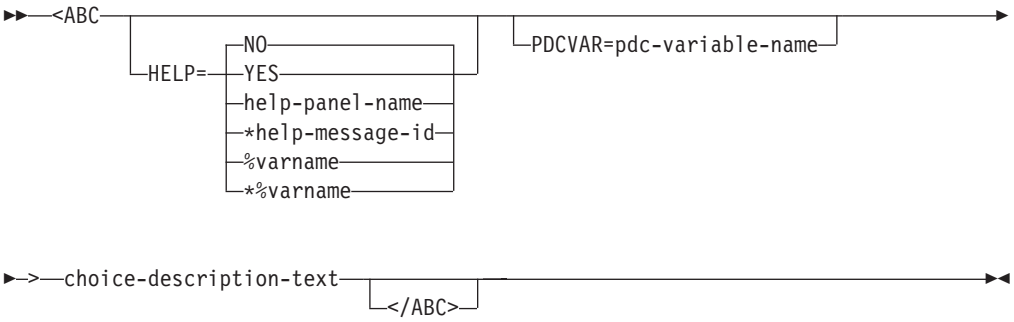
```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampbody system>)>
&sampvar1;

<PANEL NAME=ab KEYLIST=keylxmp>Library Card Registration
<AB>
<ABC>File
  <PDC>Add Entry
    <ACTION RUN=add>
  <PDC>Delete Entry
    <ACTION RUN=delete>
  <PDC>Update Entry
    <ACTION RUN=update>
  <PDC>Exit
    <ACTION RUN=exit>
<ABC>Search
  <PDC CHECKVAR=whchsrch MATCH=1>Search on name
    <ACTION SETVAR=whchsrch VALUE=1>
    <ACTION RUN=search>
  <PDC CHECKVAR=whchsrch MATCH=2>Search on card number
    <ACTION SETVAR=whchsrch VALUE=2>
    <ACTION RUN=search>
<ABC>Help
  <PDC>Extended Help...
    <ACTION RUN=exhelp>
  <PDC>Keys Help...
    <ACTION RUN=keyshelp>
</AB>
&sampbody;
</PANEL>
```

```
  File  Search  Help
------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number if applicable.

Then select an action bar choice.

Date . . . :
Card No. . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following          Check valid branches
__  1.  New                         _  North Branch
    2.  Renewal                     _  South Branch
    3.  Replacement                 _  East Branch
                                    _  West Branch

Enter a command ===> ___________________________________________________________
 F1=Help       F2=Split       F3=Exit        F6=KEYSHELP    F9=Swap
F12=Cancel
```

Figure 93. Action Bar

ABC (Action Bar Choice)

The ABC tag defines a choice in an action bar and serves as a base for associated pull-down choice tags.



HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname

This attribute specifies the help action taken when the user requests help on the action bar choice.

When HELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help on an action bar choice and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

PDCVAR=pdc-variable-name
This attribute provides the name of a variable to contain the value of the pull-down choice. When a variable name is provided, it replaces the default ZPDC variable name. The *pdc-variable-name* is not initialized to blank.

The *pdc-variable-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

choice-description-text
This is the text that appears in the action bar. The text is limited to 64 bytes.

If the *choice-description-text* exceeds the panel width, the conversion utility issues a warning message and truncates the text. If the *choice-description-text* for multiple ABC tags exceeds the panel width, the conversion utility formats a multiple-line action bar.

Description

The ABC tag defines a choice in an action bar and serves as a base for associated pull-down choice tags. The pull-down choices appear in a pull-down when the action bar choice is selected.

If the text of an action bar choice contains multiple words, multiple blanks between the words are not compressed.

Conditions

- You must code the ABC tag within an AB definition. See "AB (Action Bar)" on page 211 for a complete description of this tag.
- You must code at least one PDC tag within each ABC definition. See "PDC (Pull-Down Choice)" on page 430 for a complete description of this tag.
- The maximum number of action bar choices that will be generated is 40.

Nested Tags

You can code the following tag within an ABC definition:

Tag	Name	Usage	Page	Required
COMMENT	Comment	Multiple	282	No
M	Mnemonic	Single	392	No
PDC	Pull-down choice	Multiple	430	Yes
PDSEP	Pull-down Separator	Multiple	434	No
SOURCE	Source	Multiple	472	No

Example

The following markup illustrates the use of the PDCVAR attribute to specify an application variable for the first action bar choice. It produces the action bar on the application panel shown in Figure 94 on page 216.

ABC

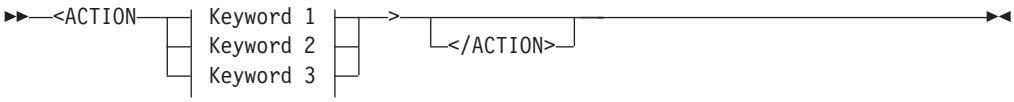
```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampbody system)>
&sampvar1;

<PANEL NAME=abc1 KEYLIST=keylxmp>Library Card Registration
<AB>
<ABC PDCVAR=foptns>File
  <PDC>Add Entry
    <ACTION RUN=add>
  <PDC>Delete Entry
    <ACTION RUN=delete>
  <PDC>Update Entry
    <ACTION RUN=update>
  <PDC>Exit
    <ACTION RUN=exit>
<ABC>Search
  <PDC CHECKVAR=whchsrch MATCH=1>Search on name
    <ACTION SETVAR=whchsrch VALUE=1>
    <ACTION RUN=search>
  <PDC CHECKVAR=whchsrch MATCH=2>Search on card number
    <ACTION SETVAR=whchsrch VALUE=2>
    <ACTION RUN=search>
<ABC>Help
  <PDC>Extended Help...
    <ACTION RUN=exhelp>
  <PDC>Keys Help...
    <ACTION RUN=keyshelp>
</AB>
&sampbody;
</PANEL>
```

```
  File  Search  Help
-------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number if applicable.

Then select an action bar choice.

Date . . . :
Card No. . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following          Check valid branches
__  1.  New                         _  North Branch
    2.  Renewal                     _  South Branch
    3.  Replacement                 _  East Branch
                                    _  West Branch

Enter a command ===> ___________________________________________________________
 F1=Help       F2=Split       F3=Exit        F6=KEYSHELP    F9=Swap
F12=Cancel
```

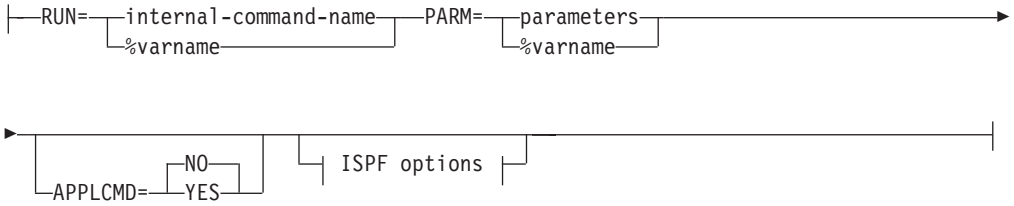
Figure 94. Action Bar Choices

ACTION (Action)

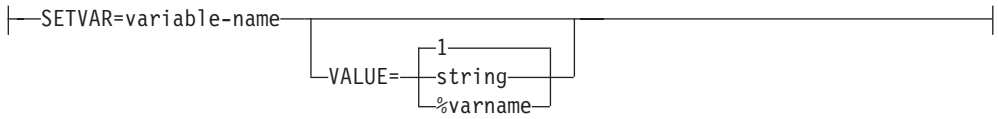
The ACTION tag defines the action that occurs when a pull-down choice or a selection field choice is selected.



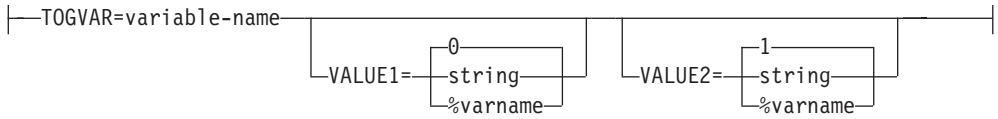
Keyword 1:



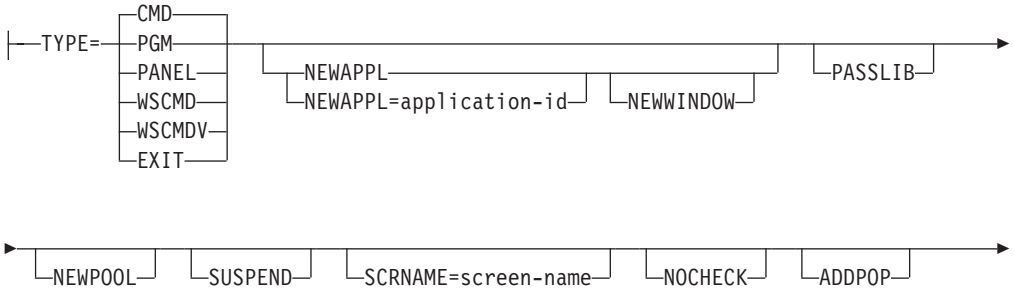
Keyword 2:



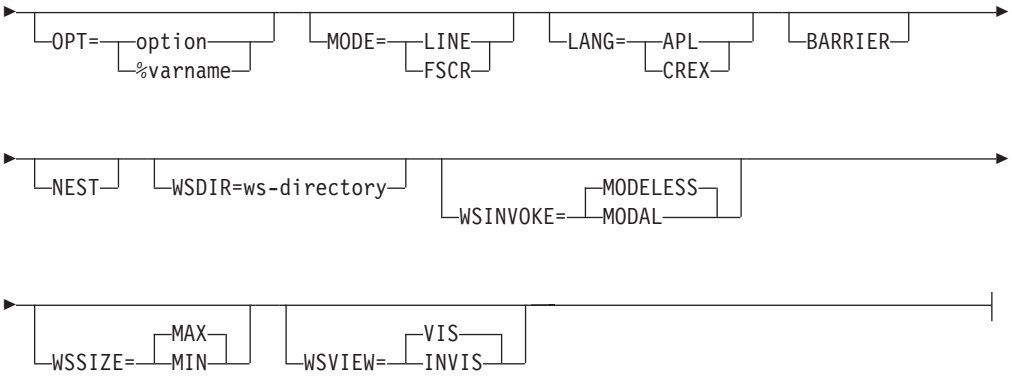
Keyword 3:



ISPF options:



ACTION



RUN=internal-command-name | %varname
When the ACTION tag is associated with a PDC tag, this attribute specifies the internal name of a command to be executed. The command is found in the application or system command table unless APPLCMD=YES is specified. The search for the command follows the normal command processing rules. For information on defining commands, see "CMD (Command Definition)" on page 268.

The RUN action is an ending action. Thus, if multiple ACTION tags are coded for a given pull-down, those following a RUN action are ignored.

When the ACTION tag is associated with a CHOICE tag (under a SELFLD tag that specifies TYPE=MENU or TYPE=MODEL), the TYPE attribute and related RUN attribute values are:

TYPE	RUN attribute value
CMD	Command name
PGM	Program name
PANEL	Panel name
WSCMD	Workstation command name and parameters
WSCMDV	The name of a variable that contains the workstation command and parameters.

When the ACTION tag is associated with a CHOICE tag under a SELFLD tag that specifies TYPE=TUTOR, the TYPE attribute is forced to ***PANEL***. The RUN attribute must provide a panel name. None of the other ISPF selection menu attributes are valid for tutorial panels.

If TYPE=CMD is specified and the *internal-command-name* should start with a %, you must code an additional % before the *internal-command-name* to distinguish it from a variable name. (For example, to specify the *internal-command-name* "%abc", code "%%abc". If TYPE=EXIT is specified, the RUN attribute is required for conversion utility processing, but is not used in the generated panel.

Note: This attribute is not supported if the ACTION tag is associated with a CHOICE tag under a SELFLD tag that specifies TYPE=SINGLE or TYPE=MULTI.

PARM=parameters | %varname
These are the command parameters. These parameters are passed to

command processing with the command specified on the RUN attribute. Command processing handles the specified parameters the same way parameters entered in the command area are handled. You can specify the name of a dialog variable (using % notation) whose value at run time will be passed as the parameter data. When the ACTION tag is associated with a PDC tag, the conversion utility limits the length of the command parameters to 72 single-byte characters.

When a ACTION tag is used to build a menu selection choice for TYPE=CMD or TYPE=PGM, and the NEWWINDOW attribute has been specified, the conversion utility limits the length of the command parameters to 249 single-byte characters; otherwise, the parameter is added to the selection as coded. The PARM attribute is not used when TYPE=WSCMD.

APPLCMD=NO | YES
This attribute specifies whether the command provided by the RUN attribute is to be passed directly to the application, bypassing the command table search. When APPLCMD=YES, the length of the command name is limited to 7 bytes to allow the passthru character ">" to be prefixed to the command name.

This attribute is valid only on an ACTION tag that is associated with a PDC tag.

The following attributes are valid only when generating an ISPF selection menu or edit model selection menu. (When the SELFLD tag specifies TYPE=TUTOR, the TYPE attribute is forced to ″PANEL″ and none of the other ISPF selection menu attributes are valid.)

TYPE=CMD | PGM | PANEL | WSCMD | WSCMDV | EXIT
This attribute specifies the type of selection to be generated for the selection menu. The attributes NEWAPPL, NEWWINDOW, PASSLIB, NEWPOOL, SUSPEND, SCRNAME, NOCHECK, ADDPOP, OPT, MODE, LANG, BARRIER, NEST, WSDIR, WSINVOKE, WSSIZE, and WSVIEW are not valid when TYPE=EXIT is specified.

NEWAPPL=application–id
The NEWAPPL keyword may be specified with or without an application identifier. This attribute specifies that the NEWAPPL keyword (and the application identifier, if present) will be added to the selection menu choice.

NEWWINDOW
This attribute specifies that the selection menu choice will be created specifying the ISPSTRT programming interface. The NEWWINDOW attribute is valid only when TYPE=PANEL, TYPE=PGM, or TYPE=CMD.

PASSLIB
This attribute specifies that the PASSLIB keyword will be added to the selection menu choice.

NEWPOOL
This attribute specifies that the NEWPOOL keyword will be added to the selection menu choice.

SUSPEND
This attribute specifies that the SUSPEND keyword will be added to the selection menu choice.

SCRNAME=screen-name
This attribute specifies that the SCRNAME keyword will be added to the selection menu choice. ISPF reserved values for *screen-name* are LIST, NEXT, PREV, ON, and OFF.

NOCHECK
This attribute specifies that the NOCHECK keyword will be added to the selection menu choice. The NOCHECK attribute is valid only when TYPE=CMD or TYPE=PGM.

ADDPOP
This attribute specifies that the ADDPOP keyword will be added to the selection menu choice. The ADDPOP attribute is valid only when TYPE=PANEL.

OPT=option | %varname
This attribute specifies that the OPT keyword will be added to the selection menu choice to specify an initial option for the panel. The OPT attribute is valid only when TYPE=PANEL.

MODE=LINE | FSCR
This attribute specifies that the MODE keyword will be added to the selection menu choice. The MODE attribute is valid only when TYPE=CMD or TYPE=PGM.

LANG=APL | CREX
This attribute specifies that the LANG keyword will be added to the selection menu choice. The LANG attribute is valid only when TYPE=CMD.

BARRIER
This attribute specifies that the BARRIER keyword will be added to the selection menu choice. The BARRIER attribute is valid only when TYPE=CMD.

NEST
This attribute specifies that the NEST keyword will be added to the selection menu choice. The NEST attribute is valid only when TYPE=CMD.

WSDIR=ws-directory
This attribute specifies that the WSDIR(*ws-directory*) keyword will be added to the selection menu choice. WSDIR provides the name of a dialog variable that contains the directory name from which the workstation command should be invoked. The WSDIR attribute is valid only when TYPE=WSCMD or TYPE=WSCMDV.

WSINVOKE=MODELESS | MODAL
This attribute specifies either the MODELESS or MODAL keyword will be added to the selection menu choice. The WSINVOKE attribute is valid only when TYPE=WSCMD or TYPE=WSCMDV.

WSSIZE=MAX | MIN
This attribute specifies either the MAX or MIN keyword will be added to the selection menu choice. The WSSIZE attribute is valid only when TYPE=WSCMD or TYPE=WSCMDV.

WSVIEW=VIS | INVIS
This attribute specifies either the VIS or INVIS keyword will be added to the selection menu choice. The WSVIEW attribute is valid only when TYPE=WSCMD or TYPE=WSCMDV.

SETVAR=variable-name
This attribute sets a value into a dialog variable. The SETVAR attribute names the variable to set. The *variable-name* must be coded without the leading % sign.

VALUE=1 | string | %varname
This is the value to set into the variable named on the SETVAR attribute. If you code the SETVAR attribute but omit the VALUE attribute, ISPF assigns the variable a value of 1. You can specify the name of a variable (using % notation) whose value at run time will be used to set the variable.

When defining the ACTION tag for selection fields, you should be aware that the variable name defined in the SELFLD tag for single-choice selection fields or in the CHOICE tag for multiple-choice selection fields will contain the value entered by the user when the selection is made. In addition, if the CHECKVAR attribute is specified in the CHOICE tag, the value of the MATCH attribute associated with the choice is set into the variable named by the CHECKVAR attribute. Therefore, it is not necessary to use the ACTION tag SETVAR attribute for the application to know which selection field choice or choices were made by the user.

TOGVAR=variable-name
This attribute allows you to alternate the value of a single variable between two values. The TOGVAR attribute names the variable to set. The *variable-name* must be coded without the leading % sign.

The function of the TOGVAR action can be depicted as follows:

```
if (TOGVAR-variable-name = VALUE1-string)
  TOGVAR-variable-name = VALUE2-string
else
  TOGVAR-variable-name = VALUE1-string
```

VALUE1=0 | string | %varname
This is the value to set into the variable named on the TOGVAR attribute if it is not currently equal to this value. If you code the TOGVAR attribute, but omit the VALUE1 attribute, the variable is assigned a value of 0. You can specify the name of a variable (using % notation) whose value at run time will be used to set the variable.

VALUE2=1 | string | %varname
This is the value to set into the variable named on the TOGVAR attribute if it is currently equal to the value specified with the VALUE1 attribute. If you code the TOGVAR attribute, but omit the VALUE2 attribute, the variable is assigned a value of 1. You can specify the name of a variable (using % notation) whose value at run time will be used to set the variable.

Description

The ACTION tag defines the action that occurs when a pull-down choice or a selection field choice is selected. Code the ACTION tag within the PDC or CHOICE definition it is associated with. You can specify multiple ACTION tags for a given choice. The conversion utility builds the logic to carry out the actions in the order in which you code the ACTION tags.

When defining action bar pull-downs, you should code the SETVAR attribute in the ACTION tags associated with each PDC tag if the application needs to know which pull-down choice the user selected. Unlike selection fields, there is no variable name associated with a pull-down definition and the PDC CHECKVAR

variable is not set to indicate the user's choice. Therefore, dialogs must refer to the SETVAR *variable-name* to determine the pull-down choice the user has selected.

The TYPE, NEWAPPL, NEWWINDOW, PASSLIB, NEWPOOL, SUSPEND, SCRNAME, NOCHECK, ADDPOP, OPT, MODE, LANG, BARRIER, NEST, WSDIR, WSINVOKE, WSSIZE, and WSVIEW attributes are used by the conversion utility to build an ISPF selection menu. They are valid only when they appear on an ACTION tag associated with a CHOICE tag which is nested within a SELFLD tag that specifies TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR (when the SELFLD tag specifies TYPE=TUTOR, the only valid selection menu attribute is TYPE=PANEL). They will not be processed in other situations. Refer to the *ISPF User's Guide* for a description of the function of these keywords in ISPF option menus.

Conditions

- You must code the ACTION tag within the PDC or CHOICE definition it is associated with. See "PDC (Pull-Down Choice)" on page 430 and "CHOICE (Selection Choice)" on page 261 for descriptions of these tags.
- You must code one (and only one) of these attributes on each ACTION tag: RUN, SETVAR, or TOGVAR.
- You can code the RUN attribute when:
  - The ACTION tag is associated with a PDC tag.
  - The ACTION tag is associated with a CHOICE tag under a SELFLD tag that specifies TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested Tags

None.

Example

In the following markup, each of the PDC tags have associated ACTION tags that specify the command that is executed when the pull-down choice is selected. Many of the PDC tags have additional ACTION tags associated with them that specify the SETVAR attribute to let the application know which pull-down choice was selected.

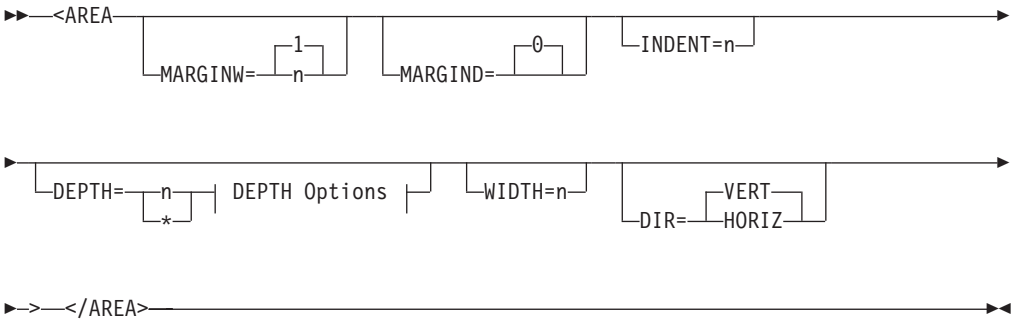
The use of ACTION tags associated with CHOICE tags is illustrated in the example for "PS (Point-and-Shoot)" on page 439.

```
<!DOCTYPE DM SYSTEM>

<PANEL NAME=action1>Library Card Listing
<AB>
<ABC>File
    <PDC>Add Entry
        <ACTION SETVAR=fchoice VALUE=add>
        <ACTION RUN=add>
    <PDC>Delete Entry
        <ACTION SETVAR=fchoice VALUE=delete>
        <ACTION RUN=delete>
    <PDC>Update Entry
        <ACTION SETVAR=fchoice VALUE=update>
        <ACTION RUN=update>
    <PDC>Exit
```

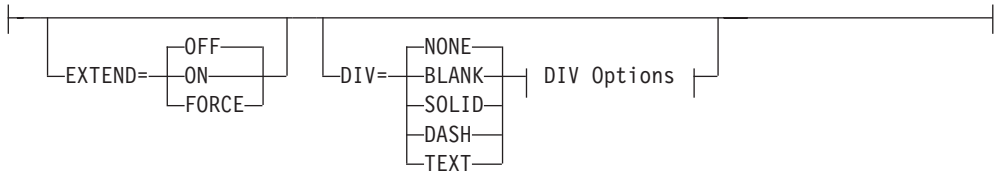
```
        <ACTION RUN=exit>
<ABC>Sort sequence
    <PDC CHECKVAR=whchsort MATCH=1>Sort on name
        <ACTION SETVAR=whchsort VALUE=1>
        <ACTION RUN=sort>
    <PDC CHECKVAR=whchsort MATCH=2>Sort on card number
        <ACTION SETVAR=whchsort VALUE=2>
        <ACTION RUN=sort>
<ABC>Help
    <PDC>Extended Help...
        <ACTION RUN=exhelp>
    <PDC>Keys Help...
        <ACTION RUN=keyshelp>
</AB>
<TOPINST>Choose the size of the list needed.
<TOPINST>Then select action bar choice "Sort sequence" to
indicate the desired sort sequence.
<AREA>
  <SELFLD NAME=aa PMTWIDTH=30 PMTLOC=before SELWIDTH=38>Choose
  one of the following
    <CHOICE>New this month
    <CHOICE>New this year
    <CHOICE>All (this will take time to process)
  </SELFLD>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

AREA (Area)

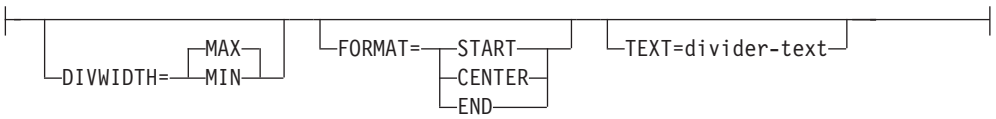
The AREA tag defines portions of a panel body, one or more of which can be scrollable.



DEPTH Options:



DIV Options:



MARGINW=1 | n
This attribute defines a margin along the left and right sides of the panel area. This attribute allows you to specify the width of the margin in characters. The minimum value you can specify is 1 and the maximum value is 32. If you do not specify a value, the margin is set to 1.

The MARGINW cannot be larger than one half the panel width minus 2. Specification of the MARGINW should always allow enough room in the panel body section of the ISPF panel being generated to contain all non-wrapped data without truncation. Specification of one half the panel width minus 2 results in no panel area in which panel body text can be written.

MARGIND=0
This attribute defines a margin along the top and bottom of the panel area.

The conversion utility will only support a margin depth of zero in an effort to use all of the available space on the panel body. Any definition of margin depth that is not equal to zero will be changed to zero.

INDENT=n
This attribute defines the number of columns to indent the current AREA from the current MARGINW value.

DEPTH=n | *
This attribute defines the minimum size of a scrollable panel area. If DEPTH is not specified for HELP panels, the conversion utility will generate multiple HELP panels for compatibility with previous releases. When EXTEND=OFF, the minimum DEPTH is 2 lines. When EXTEND=ON, the minimum DEPTH is 1 line. When DEPTH=*, the conversion utility will reserve the remaining available panel depth for the scrollable area.

EXTEND=OFF | ON | FORCE
This attribute defines the run-time display size for the scrollable area. If EXTEND=ON is specified, the panel definition is expanded from the minimum DEPTH to the size of the logical screen. Only one EXTEND=ON attribute value is allowed on a panel. The first tag (AREA, DA, GA, REGION, SELFLD) with EXTEND=ON is accepted; the EXTEND attribute on any subsequent AREA tag is ignored.

If you intend to display the panels in a pop-up window, it is recommended that you code EXTEND=OFF.

If the EXTEND attribute is specified without a DEPTH attribute, a warning message is issued and the EXTEND attribute is ignored.

If EXTEND=FORCE is specified within a horizontal area, the EXTEND(ON) keyword is added to the scrollable area attribute statement in the)ATTR panel section. The conversion utility issues a message to advise of a potential error if other panel fields are formatted on or after the last defined line of the scrollable area.

DIV=NONE | BLANK | SOLID | DASH | TEXT
This attribute specifies the type of divider line to be placed before and after the scrollable area. If this attribute is not specified, or has the value NONE, no divider line is generated. The value BLANK produces a blank

line. You must specify SOLID, DASH, or TEXT to produce a visible divider line. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line. A visible divider formats with a non-displayable attribute byte on each end of the line.

If the DIV attribute is found without the DEPTH attribute, a warning message is issued and the DIV attribute is ignored.

DIVWIDTH=MAX | MIN
This attribute specifies the width of the divider line. If DIVWIDTH=MAX, the divider line extends across the entire width of the panel defined by the AREA tag. If DIVWIDTH=MIN, the divider line is formatted to allow for the MARGINW and INDENT attribute values.

FORMAT=START | CENTER | END
This attribute specifies the position of the *divider-text* within the divider line. You must specify both the FORMAT attribute and the TEXT attribute to create a divider line containing text.

TEXT=divider-text
This attribute specifies the text to be placed on the divider line. You must specify both the FORMAT attribute and the TEXT attribute to create a divider line containing text.

WIDTH=n
This attribute defines the width of a panel area. If WIDTH is not specified the area formats to the remaining available panel width.

DIR=VERT | HORIZ
This attribute allows areas to be placed side by side on the panel. You use the WIDTH attribute in combination with the DIR attribute to tell the conversion utility to position areas horizontally. When the current horizontal AREA right boundary reaches or exceeds the right panel boundary, the next AREA will be formatted below the current AREA(s), at the left edge of the panel.

Description

The AREA tag defines portions of a panel body. The conversion utility uses the DEPTH attribute value to reserve lines in the formatted panel body for a scrollable area. The DEPTH value cannot be more than the number of panel body lines still available for formatting when the AREA tag is processed.

The maximum DEPTH that you can specify is 2 lines less than the DEPTH value specified on the HELP or PANEL tag.

Notes:

1. If you specify the CMDAREA tag within your DTL source file, it must appear before the AREA tag when DEPTH=* is specified. The AREA tag DEPTH may have to be adjusted to allow for additional lines which result from tags present within the panel definition following the end AREA tag.
2. For HELP panels, the presence of additional tags following a scrollable area causes the conversion utility to reserve 4 lines at the bottom of the screen to provide for the function key area.

The first line of the scrollable area is always reserved for the scrolling indicator line, which is provided by ISPF at run time. If all of the scrollable portion of the

panel is displayed within the available DEPTH, the scroll indicator line is blank; otherwise, the value **"More: +"**, **"More: – +"**, or **"More: –"** will appear. On application panels, this portion includes the interactive fields and text of the panel. On help panels, this portion is the area of the panel that contains help text.

The DIR attribute is used to place entire areas side by side on the panel. Formatting within the AREA tag is always in a vertical direction. Panel areas are formatted horizontally when multiple AREA tags (specifying DIR=HORIZ) are placed sequentially in the DTL source file. Any other tag which occurs between an end AREA tag and a start AREA tag causes the overall panel formatting direction to be set to vertical.

Conditions

- The AREA tag requires an end tag.
- You must code AREA tags within a HELP or PANEL definition. See "HELP (Help Panel)" on page 340 and "PANEL (Panel)" on page 415 for descriptions of these tags.
- Only one area can be defined with EXTEND=ON. This includes other AREA tags as well as any dynamic area defined by the DA tag, graphic area defined by the GA tag, scrollable section lists defined by the SELFLD tag, or scrollable regions defined by the REGION tag.

Nested Tags

Application Panel

You can code the following tags within an AREA definition on an application panel:

Tag	Name	Usage	Page	Required
COMMENT	Comment	Multiple	282	No
DA	Dynamic area	Multiple	287	No
DIVIDER	Area divider	Multiple	296	No
DTACOL	Data column	Multiple	306	No
DTAFLD	Data field	Multiple	312	No
GA *	Graphic area	Single	333	No
GENERATE	Generate	Multiple	335	No
GRPHDR	Group header	Multiple	337	No
INFO	Information region	Multiple	356	No
LSTFLD *	List field	Single	382	No
PNLINST	Panel Instruction	Multiple	437	No
REGION	Region	Multiple	445	No
SELFLD	Selection field	Multiple	455	No
SOURCE	Source	Multiple	472	No
*This tag is not valid within a scrollable area.				

Help Panel

You can code the following tag within an AREA definition on a help panel:

Tag	Name	Usage	Page	Required
COMMENT	Comment	Multiple	282	No
DIVIDER	Area divider	Multiple	296	No
GENERATE	Generate	Multiple	335	No
INFO	Information region	Multiple	356	No
REGION	Region	Multiple	445	No

Example

The application panel in the following example contains four data fields and two selection fields coded within the AREA definition. The top instructions and command area are coded outside of the AREA definition. In addition, the panels illustrate a scrollable panel. Figure 95 on page 228, Figure 96 on page 228 and Figure 97 on page 229, show the formatted results.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar2 system>
  <!entity sampabc system>)>
&sampvar2;

<PANEL NAME=area1 KEYLIST=keylxmp>File-A-Case
<AB>
&sampabc;
</AB>
<TOPINST COMPACT>
         Type in client's name and case number (if applicable).
<TOPINST>Then select an action bar choice.
<AREA>
  <DTAFLD DATAVAR=caseno PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25>Case no
    <DTAFLDD>(A 7-digit number)
  <DTAFLD DATAVAR=name PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25>Name
    <DTAFLDD>(Last, First, M.I.)
  <DTAFLD DATAVAR=address PMTWIDTH=12 ENTWIDTH=25>Address
  <DIVIDER>
  <SELFLD NAME=casesel PMTWIDTH=30 PMTLOC=before SELWIDTH=38>Choose
  one of the following
    <CHOICE CHECKVAR=case MATCH=civ>Civil
    <CHOICE CHECKVAR=case MATCH=estate>Real estate
    <CHOICE CHECKVAR=case MATCH=environ>Environmental
  </SELFLD>
</AREA>
<AREA DEPTH=6>
  <SELFLD TYPE=multi PMTWIDTH=35 SELWIDTH=50>Check type of offense committed
    <CHOICE NAME=patin HELP=patin CHECKVAR=val>Patent infringement
    <CHOICE NAME=defa HELP=defame CHECKVAR=def>Defamation
    <CHOICE NAME=cont HELP=cont CHECKVAR=qua>Breach of valid contract
    <CHOICE NAME=priv HELP=priv CHECKVAR=pri>Invasion of privacy
    <CHOICE NAME=incr HELP=incr CHECKVAR=icr>Interference with
            contractual relations
    <CHOICE NAME=disp HELP=disp CHECKVAR=dis>Improper disposal of
            medical by-products
    <CHOICE NAME=fraud HELP=fraud CHECKVAR=fra>Fraud
  </SELFLD>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

```
  File  Search  Help
------------------------------------------------------------------------------
                              File-A-Case

Type in client's name and case number (if applicable).
Then select an action bar choice.

Case no  . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following    __  1.  Civil
                                   2.  Real estate
                                   3.  Environmental
                                                          More:     +
Check type of offense committed
_  Patent infringement
_  Defamation
_  Breach of valid contract
Enter a command ===> ___________________________________________________
 F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

Figure 95. Application Panel Area

After scrolling, the panel appears as follows:

```
  File  Search  Help
------------------------------------------------------------------------------
                              File-A-Case

Type in client's name and case number (if applicable).
Then select an action bar choice.

Case no  . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following    __  1.  Civil
                                   2.  Real estate
                                   3.  Environmental
                                                          More:   - +
_  Breach of valid contract
_  Invasion of privacy
_  Interference with contractual relations
_  Improper disposal of medical by-products
Enter a command ===> ___________________________________________________
 F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

Figure 96. Application Panel Area

After scrolling, the last choice in the list is visible.

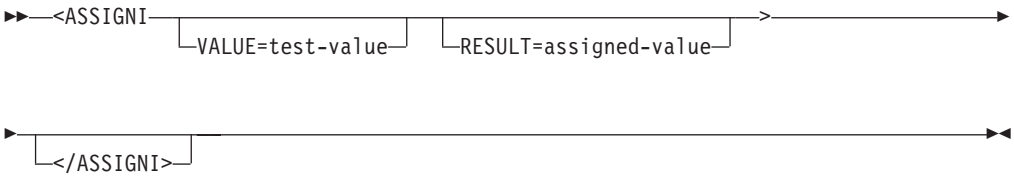
```
  File  Search  Help
------------------------------------------------------------------------------
                              File—A—Case

Type in client's name and case number (if applicable).
Then select an action bar choice.

Case no  . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following    __  1.  Civil
                                   2.  Real estate
                                   3.  Environmental
                                                          More:   -
_  Invasion of privacy
_  Interference with contractual relations
_  Improper disposal of medical by-products
_  Fraud
Enter a command ===> ____________________________________________________
 F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

Figure 97. Application Panel Area

An example of horizontal AREA formatting is shown in "Multiple AREA Tags" on page 47 .

ASSIGNI (Assignment List Item)

The ASSIGNI tag defines an assignment in an assignment list.



VALUE=test-value

This attribute specifies the value to be matched when performing the assignment.

The value of the data field variable is compared to the value of each VALUE attribute in succession until a match is found. The *test-value* must be the same case as the value to be matched. You can specify XLATL FORMAT=UPPER within the variable class associated with the data field to convert user input to uppercase before the assignment test is processed.

When ISPF finds a match, it assigns the value in the RESULT attribute to the variable named on the ASSIGNL tag. If ISPF does not find a match, no assignment is made.

If you omit this attribute, any value satisfies the test and ISPF assigns *assigned-value* to the dialog variable.

If a *test-value* appears more than once in the list, the first occurrence is used.

RESULT=assigned-value
This attribute specifies the resulting value of the assignment if a match occurs on the *test-value* specified by VALUE.

Assigned-value specifies the character string value for the conversion utility to assign to the variable named on the ASSIGNL tag. If you omit this attribute, the *test-value* is assigned to the variable.

Description

The ASSIGNI tag defines an assignment in an assignment list. Each ASSIGNI tag provides information necessary to assign a value (the RESULT attribute) to a variable (specified with the ASSIGNL tag) based on the *test-value* (the VALUE attribute) of the variable named on the DTAFLD tag. As many ASSIGNI tags as are necessary (up to a limit of 126) can be included within the assignment list. The ISPF TRANS() function is used for ASSIGNI processing.

If both the VALUE and RESULT attributes are omitted, the DESTVAR attribute of the ASSIGNL tag is assigned the value of the data field's data variable (DATAVAR).

Conditions

- You must code an ASSIGNI tag within an ASSIGNL definition. See "ASSIGNL (Assignment List)" on page 231 for a complete description of this tag.

Nested Tags

None.

Example

The following source file markup contains an application panel containing a data field. Within the data field is an assignment list that sets the dialog variable *rmtype* to 1 when "SINGLE" is entered in the data field, or to 2 when "DOUBLE" is entered in the data field. The associated variable declarations and variable classes are also shown.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=roomvar TYPE='char 6'>
  <XLATL FORMAT=upper>
  </XLATL>

<VARCLASS NAME=rmtypvar TYPE='char 1'>

<VARLIST>
  <VARDCL NAME=room   VARCLASS=roomvar>
  <VARDCL NAME=rmtype VARCLASS=rmtypvar>
</VARLIST>

<PANEL NAME=assigni DEPTH=12 WIDTH=50>Hotel Register
<AREA>
  <DTAFLD DATAVAR=room ENTWIDTH=6 DESWIDTH=20 PMTWIDTH=15>Room type
    <ASSIGNL DESTVAR=rmtype>
    <ASSIGNI VALUE=SINGLE RESULT=1>
    <ASSIGNI VALUE=DOUBLE RESULT=2>
    </ASSIGNL>
    <DTAFLDD>(Single or Double)
</AREA>
<BOTINST>Press Enter to continue.
</PANEL>
```

ASSIGNL (Assignment List)

The ASSIGNL tag defines an assignment list.

DESTVAR=destination-variable-name
DESTVAR specifies the variable that receives the assignment value. You can code multiple assignment lists if you need to assign values to additional variables.

Note: If the *destination-variable-name* is a variable name used for another field on the panel, the value of the other field will be overlaid by the assignment value. The *destination-variable-name* should only be used for an output field variable.

Description

The ASSIGNL tag defines an assignment list. ASSIGNI tags, which define the elements of the assignment list, are coded within the ASSIGNL tag.

Assignment lists are optional and provide a means of assigning a value to one variable based on the content of another. ISPF compares the value of the variable specified with the DATAVAR attribute of the DTAFLD tag against the values in the ASSIGNI tags.

Processing of assignment lists occurs at panel end (after translates and checks).

Conditions

- The ASSIGNL tag requires an end tag.
- You must code an ASSIGNL tag within the DTAFLD definition it is associated with. See "DTAFLD (Data Field)" on page 312 for a complete description of this tag.

Nested Tags

You can code the following tag within an ASSIGNL definition:

Tag	Name	Usage	Page	Required
ASSIGNI	Assignment list item	Multiple	229	No

Example

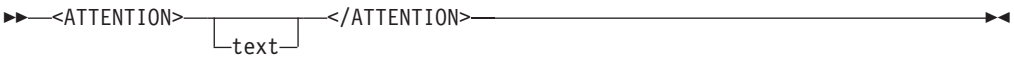
The assignment list in the following markup assigns a value to the variable *decimal* dependent on the value the user enters in the data field variable *hexvar*. The associated variable declarations and variable classes are also shown.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=varcls1 TYPE='char 1'>

<VARCLASS NAME=varcls2 TYPE='char 2'>

<VARLIST>
  <VARDCL NAME=hexvar  VARCLASS=varcls1>
  <VARDCL NAME=decimal VARCLASS=varcls2>
```

```
</VARLIST>

<PANEL NAME=assignl>Hex to Decimal
<TOPINST>Enter a hexadecimal digit.
<AREA>
<DTAFLD DATAVAR=hexvar PMTWIDTH=23 ENTWIDTH=1>Hexadecimal Value
  <ASSIGNL DESTVAR=decimal>
    <ASSIGNI VALUE=0>
    <ASSIGNI VALUE=1>
    <ASSIGNI VALUE=2>
    <ASSIGNI VALUE=3>
    <ASSIGNI VALUE=4>
    <ASSIGNI VALUE=5>
    <ASSIGNI VALUE=6>
    <ASSIGNI VALUE=7>
    <ASSIGNI VALUE=8>
    <ASSIGNI VALUE=9>
    <ASSIGNI VALUE=a RESULT=10>
    <ASSIGNI VALUE=b RESULT=11>
    <ASSIGNI VALUE=c RESULT=12>
    <ASSIGNI VALUE=d RESULT=13>
    <ASSIGNI VALUE=e RESULT=14>
    <ASSIGNI VALUE=f RESULT=15>
    <ASSIGNI RESULT="??">
  </ASSIGNL>
<DTAFLD DATAVAR=decimal USAGE=out PMTWIDTH=23 ENTWIDTH=2>Decimal Value
</AREA>
</PANEL>
```

ATTENTION (Attention)

The ATTENTION tag defines text that alerts the user to a risk of possible error conditions in the system.



text
This is the text of the attention message.

Description

The ATTENTION tag defines text that alerts the user to a risk of possible error conditions in the system.

The ATTENTION tag is one of the tags that alert the user of a possible risk; CAUTION and WARNING are the others.

Code an attention statement before the text to which it pertains so that the user can read about the possible risks before reading the text.

When an attention statement is displayed, the string “Attention:” (or its translated equivalent) appears on the screen before the text of the statement.

You can code additional paragraphs of text by coding the P (paragraph) tag within an ATTENTION definition. You can also code other tags allowed in an information area within an ATTENTION definition.

Conditions

- The ATTENTION tag requires an end tag.
- You must code the ATTENTION tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.
- The ATTENTION tag must be immediately preceded by a P, LI, or LP tag. If the ATTENTION tag is coded on the same line as one of these tags, there can be no intervening blanks. See "P (Paragraph)" on page 408, "LI (List Item)" on page 364, and "LP (List Part)" on page 370 for descriptions of these tags.
- You cannot nest ATTENTION, WARNING or CAUTION tags within each other.

Nested Tags

You can code the following tags within an ATTENTION definition:

Tag	Name	Usage	Page	Required
DL	Definition list	Multiple	298	No
FIG	Figure	Multiple	329	No
HP	Highlighted phrase	Multiple	354	No
LINES	Lines	Multiple	366	No
NOTE	Note	Multiple	400	No
NOTEL	Note List	Multiple	402	No
NT	Note	Multiple	404	No
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No
XMP	Example	Multiple	497	No

Example

The following help panel markup contains a warning statement. The warning statement starts at the left margin because it is imbedded in the LP tag.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=attentn DEPTH=20>Help For Changing a File
<AREA>
<INFO>
  <OL>
    <LI>Type over the existing data
    in the entry fields with the new data.
      <LP><ATTENTION>Performing the next step will save
      all changes and delete the existing data.
      <P>To quit this function without
      deleting the existing data, press F12.
      </ATTENTION>
    <LI>Press Enter to save the
    updated data.
```

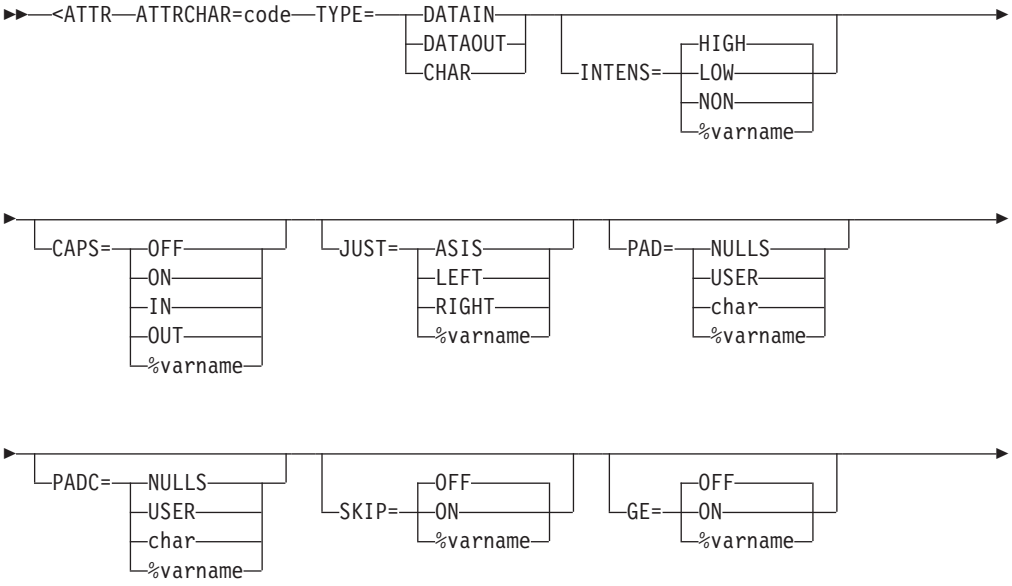
```
  </OL>
</INFO>
</AREA>
</HELP>
```

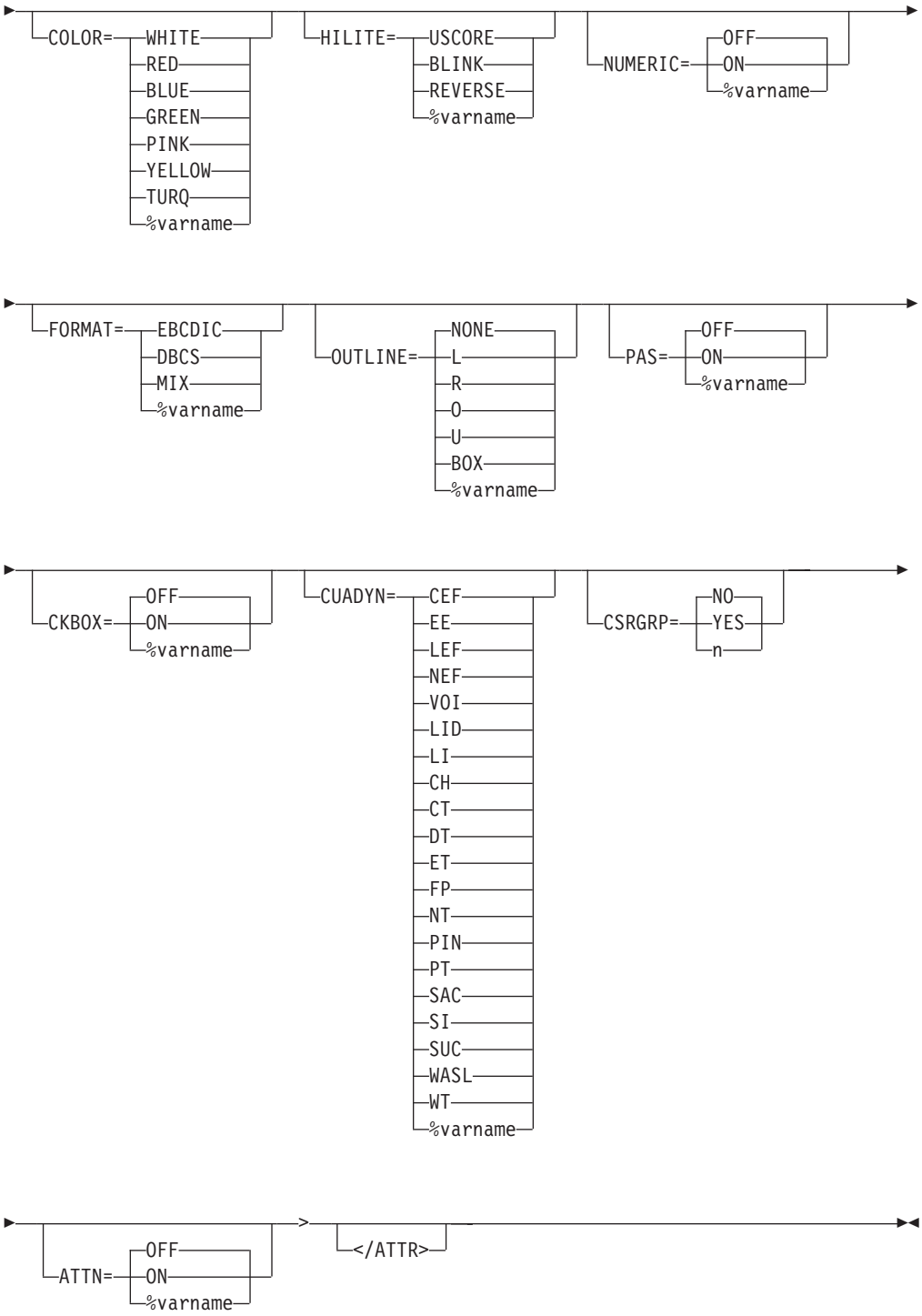
```
            Help For Changing a File

1.  Type over the existing data in the entry
    fields with the new data.

Attention: Performing the next step will save all
changes and delete the existing data.

To quit this function without deleting the
existing data, press F12.

2.  Press Enter to save the updated data.




 F1=Help          F3=Exit          F5=Exhelp
 F6=Keyshelp      F7=PrvTopic      F8=NxtTopic
F10=PrvPage      F11=NxtPage      F12=Cancel
```

Figure 98. Attention Statement

ATTR (Attribute)

The ATTR tag defines a panel attribute used within a dynamic area or a preformatted displayable panel section. Refer to the *ISPF User's Guide* for a complete discussion of panel)ATTR section keywords.





ATTRCHAR=code

This attribute can be a single character or a two-position entry of valid hex digits. If you enter an incorrect value, a warning message is issued and the value is set to null. Hex entries are converted to character. Hex values '00'–'2F' are reserved for use by the conversion utility.

TYPE=DATAIN | DATAOUT | CHAR

This attribute specifies the characteristic of the field within the dynamic area. Use DATAIN and DATAOUT attribute values for specifying unprotected or

protected fields, respectively, within the dynamic area. The CHAR attribute value defines a character attribute that is recognized only when found within a shadow variable.

When the ATTR tag is coded within the GENERATE tag, TYPE can also be specified as any CUA attribute type, or as %varname.

INTENS=HIGH | LOW | NON | %varname
This attribute defines the intensity of a field. You can define this attribute as a variable name preceded by a "%".

CAPS=OFF | ON | IN | OUT | %varname
This attribute specifies the uppercase or lowercase attribute of a field. You can define this attribute as a variable name preceded by a "%".

JUST=ASIS | LEFT | RIGHT | %varname
This attribute specifies how the contents of the field are to be justified when displayed. You can define this attribute as a variable name preceded by a "%".

PAD=NULLS | USER | char | %varname
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

PADC=NULLS | USER | char | %varname
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

SKIP=OFF | ON | %varname
This attribute specifies the autoskip attribute of the field. You can define this attribute as a variable name preceded by a "%".

GE=OFF | ON | %varname
This attribute specifies that ISPF will place a graphic escape order before the character defined as a character level attribute by TYPE=CHAR. You can define this attribute as a variable name preceded by a "%".

COLOR=WHITE | RED | BLUE | GREEN | PINK | YELLOW | TURQ | %varname
This attribute specifies the color of the field. You can define this attribute as a variable name preceded by a "%".

HILITE=USCORE | BLINK | REVERSE | %varname
This attribute specifies the extended highlighting attribute of the field. You can define this attribute as a variable name preceded by a "%".

NUMERIC=OFF | ON | %varname
This attribute specifies whether Numeric Lock is to be activated for data typed in the field. You can define this attribute as a variable name preceded by a "%".

FORMAT=EBCDIC | DBCS | MIX | %varname
This attribute specifies the character format for the field. You can define this attribute as a variable name preceded by a "%".

OUTLINE=NONE | L | R | O | U | BOX | %varname
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

PAS=OFF | ON | %varname
This attribute controls the availability of the point-and-shoot function for dynamic areas. You can define this attribute as a variable name preceded by a "%".

CKBOX=OFF | ON | %varname
This attribute controls the generation of check boxes for dynamic areas when the panel is displayed while running in GUI mode. You can define this attribute as a variable name preceded by "%".

CUADYN=CEF | EE | LEF | NEF | VOI | LID | LI | CH | CT | DT | ET | FP | NT | PIN | PT | SAC | SI | SUC | WASL | WT | %varname
This attribute specifies a standard CUA attribute for the DATAIN and DATAOUT panel attribute definitions.

Values CEF, EE, LEF, and NEF are valid when TYPE=DATAIN. The remaining values are valid when TYPE=DATAOUT. You can define this attribute as a variable name preceded by a "%".

The conversion utility will issue a warning message if the CUADYN attribute is specified and the invocation option is NOCUAATTR.

CSRGRP=NO | YES | n
The CSRGRP attribute is valid only when TYPE=DATAOUT. When CSRGRP=YES, the conversion utility generates a cursor group number to be used for this DATAOUT attribute. When CSRGRP=n, the number provided is used for this attribute. The PAS attribute must be specified as ON or %varname.

ATTN=NO | YES | %varname
This attribute specifies the attention-select attribute of the field. You can define this attribute as a variable name preceded by a "%".

Description

The ATTR tag is used to create an entry in the)ATTR panel section for fields to be displayed within a dynamic area, or preformatted panel section.

Conditions

- You must code an ATTR tag within a Dynamic Area or GENERATE tag definition. See "DA (Dynamic Area)" on page 287 and "GENERATE (Generate)" on page 335 for a complete description of these tags.
- If ATTRCHAR is not specified, an error is logged and the output panel is not saved.
- If ATTRCHAR is a duplicate of a previously specified attribute, or conflicts with an attribute reserved by the conversion utility, an error is logged and the output panel is not saved.
- If TYPE is not specified, an error is logged and the output panel is not saved. If TYPE is specified, but the value is invalid, the value is set to DATAIN.
- If both PAD and PADC have been specified, PAD is ignored and PADC is used.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested Tags

None.

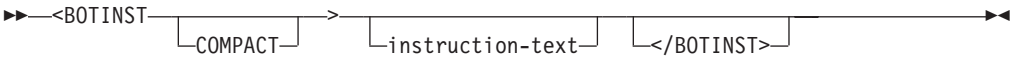
Example

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampabc system>)>
&sampvar1;
```

```
<PANEL NAME=attr KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
<TOPINST> Type in patron's name and card number (if applicable)
<AREA>
  <DTACOL PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25 SELWIDTH=25>
    <DTAFLD DATAVAR=curdate USAGE=out ENTWIDTH=8>Date
    <DTAFLD DATAVAR=cardno ENTWIDTH=7>Card No.
      <DTAFLDD>(A 7-digit number)
    <DTAFLD DATAVAR=name>Name
      <DTAFLDD>(Last, First, M.I.)
    <DTAFLD DATAVAR=address>Address
  </DTACOL>
  <DIVIDER>
  <DA NAME=darea DIV=solid DEPTH=6 SHADOW=shadwvar>
    <ATTR ATTRCHAR=#  TYPE=datain    PADC='_'  COLOR=BLUE>
    <ATTR ATTRCHAR=|  TYPE=dataout   COLOR=green>
    <ATTR ATTRCHAR=$  TYPE=char      COLOR=red>
  </DA>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

BOTINST (Bottom Instruction)

The BOTINST tag defines bottom instructions for an application panel.



COMPACT
This attribute causes the bottom instruction to format without a blank line before the text.

instruction-text
This is the text of the bottom instruction. The *instruction-text* must fit in the remaining panel depth.

Description

The BOTINST tag defines bottom instructions for an application panel. The *instruction-text* formats as a paragraph based on the width of the application panel. You can code multiple paragraphs of instruction text by using a new bottom instruction tag for each new paragraph.

If the COMPACT attribute is not specified, the conversion utility inserts a blank line before the bottom instruction text.

Conditions

- You must code the BOTINST within a PANEL definition. See "PANEL (Panel)" on page 415 for a complete description of this tag.
- You cannot code a BOTINST tag within an AREA definition. If you define an area for the panel, code the BOTINST tag after the AREA end tag.

Nested Tags

You can code the following tags within a BOTINST definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

The following application panel markup contains two bottom instructions. Figure 99 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=choiccls TYPE='char 2'>
<VARLIST>
  <VARDCL NAME=choices VARCLASS=choiccls>
</VARLIST>

<PANEL NAME=botinst1 WIDTH=35 DEPTH=22>Four Choices
<AREA>
  <SELFLD NAME=choices PMTWIDTH=20 SELWIDTH=33>Choose one:
    <CHOICE>Raindrops on roses
    <CHOICE>Whiskers on kittens
    <CHOICE>Bright copper kettles
    <CHOICE>Warm woolen mittens
  </SELFLD>
</AREA>
<BOTINST>Press Enter to continue.
<BOTINST>Press F12 to cancel.
</PANEL>
```

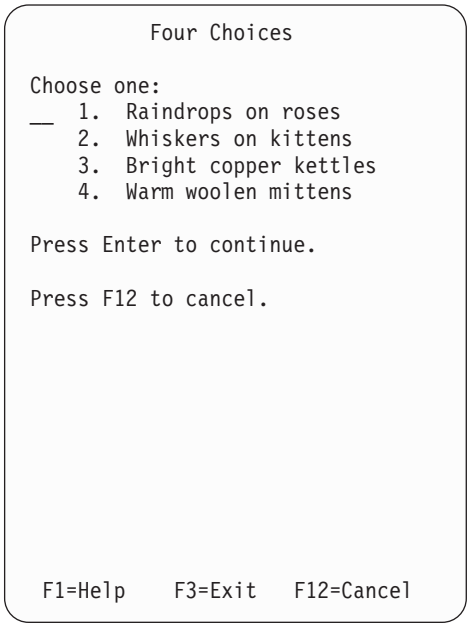


Figure 99. Bottom Instructions

CAUTION (Caution)

The CAUTION tag defines a statement that alerts the user of a possible risk.

text
This is the text of the caution statement.

Description

The CAUTION tag defines a statement that alerts the user of a possible risk. Use the CAUTION tag to alert the user to a condition that might have serious consequences, such as the deletion of a file.

The CAUTION tag is one of the tags that alert the user to a possible risk; the others are ATTENTION and WARNING.

Code a caution statement before the text to which it pertains so that the user can read about the possible risks before reading the text. When a caution statement is displayed, the string "CAUTION:" or its translated equivalent appears on the screen and the caution text displays on the following line.

You can code additional paragraphs of caution text by coding the P (paragraph) tag within a CAUTION definition. You can also code other tags allowed in an information area within a CAUTION definition.

CAUTION text is formatted with an attribute byte that causes it to be emphasized.

Conditions

- The CAUTION tag requires an end tag.
- A CAUTION tag must be immediately preceded by an LI, LP, or P tag. See "LI (List Item)" on page 364, "LP (List Part)" on page 370, and "P (Paragraph)" on page 408 for descriptions of these tags.
- You must code the CAUTION tag only within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.
- You cannot nest ATTENTION, CAUTION, or WARNING tags within each other.

Nested Tags

You can code the following tags within a CAUTION definition:

Tag	Name	Usage	Page	Required
DL	Definition list	Multiple	298	No
FIG	Figure	Multiple	329	No
HP	Highlighted phrase	Multiple	354	No
LINES	Lines	Multiple	366	No
NOTE	Note	Multiple	400	No
NOTEL	Note list	Multiple	402	No
NT	Note	Multiple	404	No

Tag	Name	Usage	Page	Required
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No
XMP	Example	Multiple	497	No

Example

The following help panel markup contains a caution statement. Figure 100 shows the formatted result.

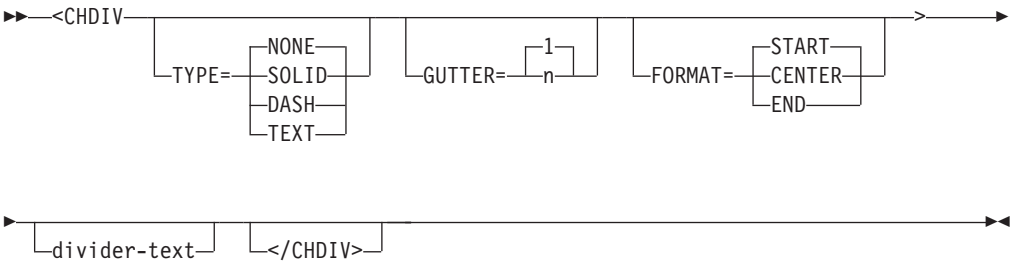
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=caution DEPTH=20>Help for DELETE Command
<AREA>
  <INFO>
    <P>The DELETE command erases the specified file from storage.
    <P><CAUTION>
    Issuing the DELETE command permanently
    removes the file from storage.
    There is no possibility of recovery.
    </CAUTION>
    <P>You can exit from the DELETE operation
    by pressing F12.
  </INFO>
</AREA>
</HELP>
```

```
             Help for DELETE Command

 The DELETE command erases the specified file
 from storage.

 CAUTION:
 Issuing the DELETE command permanently removes
 the file from storage. There is no possibility
 of recovery.

 You can exit from the DELETE operation by
 pressing F12.




  F1=Help         F3=Exit         F5=Exhelp
  F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
 F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 100. Caution Statement

CHDIV (Choice Divider)

The CHDIV tag creates a blank or visible divider between CHOICE tags.



TYPE=NONE | SOLID | DASH | TEXT
This attribute specifies the type of divider line. The line width is one character.

The default value is *none*, which produces a blank line. You must specify *solid*, *dash*, or *text* to produce a visible divider line. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line.

GUTTER=1 | n
This attribute specifies the total width of the divider. If the GUTTER value is an even number, the conversion utility increases the number by 1 so that the divider is centered within the defined width.

The minimum and default GUTTER value is 1.

FORMAT=START | CENTER | END
This attribute specifies the position of the divider text within the width of the divider line. The divider width is the same as the CHOICE tag text formatting width.

divider-text
This is the text of the choice divider.

Description

The CHDIV tag creates a blank or solid divider between CHOICE tags.

Conditions

- You must code the CHDIV tag within an SELFLD definition. See "SELFLD (Selection Field)" on page 455 for a description of this tag.

Nested Tags

You can code the following tags within a CHDIV definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No

Example

The following example illustrates the creation of an ISPF selection menu. The CHDIV tag is included to separate the Exit option from the other selection choices.

The FCHOICE attribute specifies that the first selection number is 0. The choice selection for Exit is specified on the CHOICE tag. The ACTION tag for the Exit choice selection specifies both the RUN and TYPE attributes because RUN is required on the ACTION tag and TYPE is necessary to specify the ISPF selection for the generated ZSEL panel statement.

```
<!doctype dm system ()>
<!--  Sample selection menu -->
<varclass name=vc1 type='char 80'>
  <xlatl format=upper>
  </xlatl>

<varlist>
  <vardcl name=zcmd varclass=vc1>
</varlist>

<panel name=chdiv1 menu keylist=keylxmp>
       Sample Selection Panel with CHDIV tag
  <topinst>This is a selection panel.
  <selfld type=menu   pmtloc=before fchoice=0 trail=nextsel
          selwidth=40 pmtwidth=10>Select an option
    <choice checkvar=xtest1 match=a>
            Selection #0 (Command Selch0)
      <action run=Selch0>
    <choice checkvar=xtest1 match=b>
            Selection #1 (Command Selch1)
      <action run=Selch1 parm='1 2 3 4'
       passlib newpool suspend>
    <choice checkvar=xtest1 match=c>
            Selection #2 (Command Selch2)
      <action run=Selch2 parm=1234>
    <choice checkvar=xtest1 match=d>
            Selection #3 (Command Selch3)
      <action run=Selch3 parm=abcd>
    <choice checkvar=xtest1 match=e>
            Selection #4 (Command Selch4)
      <action run=Selch4 parm='a b c d'>
    <chdiv>
    <choice selchar=X>
            Exit
      <action run=exit type=exit>
  </selfld>
  <cmdarea>
</panel>
```

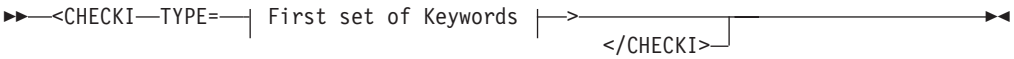
Figure 101 on page 244 shows the formatted result.

```
                  Sample Selection Panel with CHDIV tag
Option ===> ____________________________________________________________

This is a selection panel.

Select an
option . . 0  Selection #0 (Command Selch0)
           1  Selection #1 (Command Selch1)
           2  Selection #2 (Command Selch2)
           3  Selection #3 (Command Selch3)
           4  Selection #4 (Command Selch4)

           X  Exit









Option ===> ____________________________________________________________
 F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

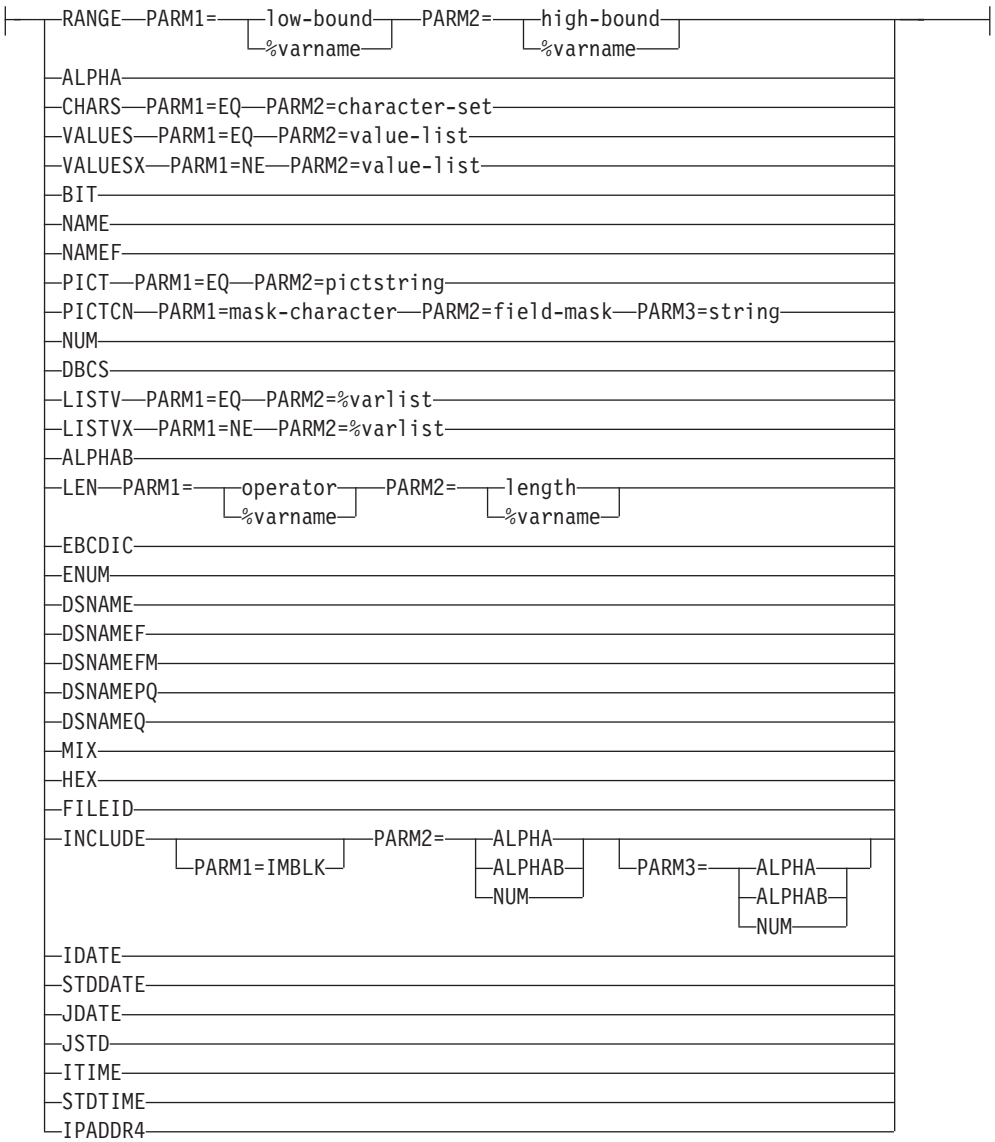
Figure 101. Choice Divider

CHECKI (Validity Check Item)

The CHECKI tag defines a test of an input value.



First set of Keywords



TYPE=

This attribute specifies the type of check to be performed. The valid types are:

RANGE

This allows you to check for an integer value within a range. The specified range includes the end points. The range delimiters can include 16 digits. The range delimiters can also contain a sign (– or +). If no sign is coded, the value is assumed to be positive.

Attention:

In ISPF, the VER(variable RANGE,lower,upper) statement limits the length of the specified variable to 16 digits. If the application developer specifies the CHECKI TYPE=RANGE on a variable that is longer than 16 positions in length, the variable may not be correctly validated.

For example, assume an application developer defines a data field with a length of 20 and defines the following validity check for the field:

```
<CHECKI TYPE=RANGE PARM1=1 PARM2=9999999999999999>
```

If the number `12345678901234567890` were entered into the field, only the first 16 digits of the field would be verified and the number would be within the defined range, even though the entire number entered is outside of the defined range.

PARM1=low-bound | %varname
This attribute supplies the low value, if any or the name of a variable that contains the low value. If you do not supply a value, the default is "–" followed by sixteen 9s (that is, –9999...99). Negative values must be coded with the minus sign on the left.

ISPF restrictions on the VER(variable RANGE,lower,upper) apply. The lower value specified in PARM1 can be positive or negative. The length of the lower limit is limited to 16 digits, in addition to the plus or minus sign if used.

PARM2=high-bound | %varname
This attribute supplies the high value, if any or the name of a variable that contains the high value. If you do not supply a value, the default is sixteen 9s (that is, 9999...99). Negative values must be coded with the minus sign on the left.

ISPF restrictions on the VER(variable RANGE,lower,upper) apply. The upper value specified in PARM2 can be positive or negative. The length of the upper limit is limited to 16 digits, in addition to the plus or minus sign if used.

ALPHA
This limits the character set to A–Z, a–z, and #, $, @. The conversion utility builds the VER(variable ALPHA) statement.

CHARS
Specifies the CHARS check of characters allowed within an input string.

The conversion utility uses the TYPE=CHARS check to support ISPF VER(variable BIT), VER(variable HEX) and VER(variable NUM). BIT, HEX, and NUM are the only types of support provided by ISPF for the TYPE=CHARS check.

PARM1=EQ
This attribute contains EQ to specify that PARM2 contains a value that must be matched. If PARM1 contains any other value, the check is ignored and the conversion utility issues a warning message.

PARM2=character-set
This attribute specifies a set of characters to be matched.

- If TYPE=CHARS, PARM1='EQ', and PARM2='01', the conversion utility generates VER(variable BIT).

- If TYPE=CHARS, PARM1='EQ', and PARM2='0123456789ABCDEFabcdef', the conversion utility generates VER(variable HEX).
- If TYPE=CHARS, PARM1='EQ', and PARM2='0123456789', the conversion utility generates VER(variable NUM).

  Note: If these options are used, the PARM2 value must be specified exactly as shown above.
- If PARM2 contains any other values, the check is ignored and the conversion utility issues a warning message.

VALUES

Specifies that the value entered must be the same as one of the values specified in PARM2.

The VER LIST statement built by the conversion utility is case-sensitive to the values entered in PARM2 (no folding of PARM2 to uppercase). This means that ISPF looks for an exact match in the verification process. You can specify XLATL FORMAT=UPPER within the variable class definition before the CHECKL tag to convert user input to uppercase before the VALUES check is processed.

PARM1=EQ

This attribute contains EQ to specify that PARM2 contains values that must be matched. If PARM1 contains any other value, the check is ignored and the conversion utility issues a warning message.

PARM2=value-list

This attribute specifies the list of values. If the list contains more than one value, it must be enclosed in quotes. If a value in the list contains blanks, then it must be enclosed in single quotes and the entire list enclosed in double quotes. Each value in the list must be separated by blanks or enclosed quotes. For example:

```
dog
'dog cat bird lion'
"parsley onion 'black pepper' garlic"
```

The maximum number of values allowed is 100.

Note: You should surround the entire value for PARM2 with double quotes and then surround any value in the list that contains blanks with single quotes. Double quotes surrounding a list are supported by the conversion utility.

The conversion utility generates VER(variable LIST,value-list) from PARM2.

VALUESX

Specifies that the value entered cannot be any of the values specified in PARM2. This is the opposite of VALUES.

The VER LISTX statement built by the conversion utility is case-sensitive to the values entered in PARM2 (no folding of PARM2 to uppercase). This means that ISPF looks for an exact match in the verification process. You can specify XLATL FORMAT=UPPER within the variable class definition before the CHECKL tag to convert user input to uppercase before the VALUES check is processed.

PARM1=NE

This attribute contains NE to specify that PARM2 contains values that

cannot be entered. If PARM1 contains any other value, the check is ignored and the conversion utility issues a warning message.

PARM2=VALUE-LIST
This attribute specifies the list of values. If the list contains more than one value, it must be enclosed in quotes. If a value in the list contains blanks, then it must be enclosed in single quotes and the entire list enclosed in double quotes. Each value in the list must be separated by blanks or enclosed quotes. For example:

```
dog
'dog cat bird lion'
"parsley onion 'black pepper' garlic"
```

The maximum number of values allowed is 100.

Note: You should surround the entire value for PARM2 with double quotes and then surround any value in the list that contains blanks with single quotes. Double quotes surrounding a list are supported by the conversion utility.

The conversion utility generates VER(variable LISTX,value-list) from PARM2.

BIT
Specifies that the variable must be all 0's and 1's. The conversion utility builds the VER(variable BIT) statement.

NAME
Specifies that the variable must contain a valid name, following the rules of member names. The conversion utility builds the VER(variable NAME) statement.

NAMEF
Specifies that the variable must contain a valid name filter. The conversion utility builds the VER(variable NAMEF) statement.

PICT
Specifies that the variable must contain characters that match the corresponding type of character in *pictstring*.

PARM1=EQ
This attribute contains EQ to specify that PARM2 contains values that must be matched. If PARM1 contains any other value, the check is ignored and the conversion utility issues a warning message.

PARM2=pictstring
This 'pictstring' parameter must be the actual value to be used in the generated VER statement. ISPF does not support a variable name for this value.

If PARM2 contains invalid characters as defined by ISPF, the check is ignored and the conversion utility issues a warning message.

The conversion utility builds the VER(variable PICT,pictstring) statement.

PICTCN
Specifies that the variable must contain specific constants along with other standard ISPF picture verification characters.

PARM1=mask-character
The mask-character is any special character that represents itself. It cannot be one of the ISPF picture string characters (C,A,N,X,9,c,a,n,x)

PARM2=field-mask
The field-mask provides the required characters for the field. All other field positions must be represented by the mask-character. For example, if the field is to contain a string **VnnRnnMnn** (for Version, Release, and Modification) and the mask-character is an asterisk (*), the field mask is V**R**M**.

PARM3=string
The string must be the same length as the field-mask. It contains all of the required characters in the same positions as the field-mask. The positions defined with the mask-character in the field-mask contain one of the standard ISPF picture characters (C,A,N,X,9,c,a,n,x). To complete the example used above, the string is VnnRnnMnn. The resulting verification statement is:

```
VER(variable,PICTCN,*,V**R**M**,VnnRnnMnn)
```

The variable is verified for **V** in position 1, **R** in position 4, **M** in position 7, and numeric characters in positions 2,3,5,6,8, and 9.

The conversion utility builds the VER(variable,PICTCN,mask-character,field-mask,string) statement.

NUM
Specifies that the variable must contain all numeric characters (0–9). The conversion utility builds the VER(variable NUM) statement.

DBCS
Specifies that the variable must contain all valid DBCS characters. The conversion utility builds the VER(variable DBCS) statement.

LISTV
Specifies that the variable must be one of the values provided by *varlist*.

PARM1=EQ
This attribute contains EQ to specify that PARM2 contains values that must be matched. If PARM1 contains any other value, the check is ignored and the conversion utility issues a warning message.

PARM2=%varlist
This attribute must be a variable name entered with "%" as the first character. The variable contents are provided by the application and must be a list of valid values.

The conversion utility builds the VER(variable LISTV,&varlist) statement.

LISTVX
Specifies that the variable cannot be any of the values provided by *varlist*. This is the opposite of LISTV.

PARM1=NE
This attribute contains NE to specify that PARM2 contains values that cannot be entered. If PARM1 contains any other value, the check is ignored and the conversion utility issues a warning message.

PARM2=%VARLIST
This attribute must be a variable name entered with "%" as the first

character. The variable contents are provided by the application and must be a valid list of excluded values.

The conversion utility builds the VER(variable LISTVX,&varlist) statement.

ALPHAB
Specifies that the variable must be all alphabetic characters (A–Z or a–z). The conversion utility builds the VER(variable ALPHAB) statement.

LEN
Specifies that the variable must satisfy the condition expressed by the relational operator and the expected length.

PARM1=operator | %varname
This attribute can be a relational operator (EQ, LT, GT, LE, GE, NE, NG, or NL) or a variable name that contains a relational operator. If a variable name is entered, it must be preceded by a "%".

PARM2=length | %varname
The parameter must be either a number or a variable name. If a number is entered, it must be in the range of 1–99999. If a variable name is entered, it must be preceded by a "%".

The conversion utility builds the VER(variable operator,length) statement.

EBCDIC
Specifies that the variable must contain all valid EBCDIC characters. The conversion utility builds the VER(variable EBDIC) statement.

ENUM
Specifies that the variable can contain extended numeric notation. The conversion utility builds the VER(variable ENUM) statement.

DSNAME
Specifies that the variable must contain a valid TSO data set name. The conversion utility builds the VER(variable DSNAME) statement.

DSNAMEF
Specifies that the variable must contain a valid TSO data set name filter. The optional member name cannot be specified as a member pattern. A missing close parentheses and ending quotation mark are automatically added by ISPF. The conversion utility builds the VER(variable DSNAMEF) statement.

DSNAMEFM
Specifies that the variable must contain a valid data set name. The optional member name can be specified as a member pattern. A missing close parentheses and ending quotation mark are automatically added by ISPF. The conversion utility builds the VER(variable DSNAMEFM) statement.

DSNAMEPQ
Specifies that the variable must contain a valid TSO data set name. A missing close parentheses and ending quotation mark are automatically added by ISPF. The conversion utility builds the VER(variable DSNAMEPQ) statement.

DSNAMEQ
Specifies that the variable must contain a valid TSO data set name. A missing ending quotation mark is automatically added by ISPF. The conversion utility builds the VER(variable DSNAMEQ) statement.

MIX
Specifies that the variable must contain all valid DBCS and EBCDIC characters. The conversion utility builds the VER(variable MIX) statement.

HEX
Specifies that the variable must contain all hexadecimal characters (0–9, a–f or A–F). The conversion utility builds the VER(variable HEX) statement.

FILEID
Specifies that the variable must contain a valid file ID in CMS syntax. The conversion utility builds the VER(variable FILEID) statement.

Refer to the *ISPF User's Guide* for additional information concerning panel variable validation.

INCLUDE
Specifies that the variable must contain valid characters from at least one of the ISPF defined VER groups ALPHA, ALPHAB or NUM.

PARM1=IMBLK
This attribute contains IMBLK to specify that the IMBLK keyword be added to the generated VER statement. If PARM1 contains any other value, it will be reset to the value IMBLK.

PARM2=ALPHA | ALPHAB | NUM
This attribute must contain either the value ALPHA, ALPHAB, or NUM. If PARM2 is not specified or contains any other value, the INCLUDE check is ignored and the conversion utility issues a warning message.

PARM3=ALPHA | ALPHAB | NUM
This attribute must contain either the value ALPHA, ALPHAB, or NUM. The value specified for PARM3 should be different than the value specified for PARM2. If the values for PARM2 and PARM3 are the same, the PARM3 value is ignored and the conversion utility issues a warning message.

The conversion utility builds the VER(variable INCLUDE [,IMBLK], parm2 [,parm3]) statement.

IDATE
Specifies that the variable must contain a valid 8 character international date. The conversion utility builds the VER(variable,IDATE) statement.

STDDATE
Specifies that the variable must contain a valid 10 character standard date. The conversion utility builds the VER(variable,STDDATE) statement.

JDATE
Specifies that the variable must contain a valid 6 character Julian date. The conversion utility builds the VER(variable,JDATE) statement.

JSTD
Specifies that the variable must contain a valid 8 character standard Julian date. The conversion utility builds the VER(variable,JSTD) statement.

ITIME
Specifies that the variable must contain a valid 5 character international time. The conversion utility builds the VER(variable,ITIME) statement.

STDTIME
Specifies that the variable must contain a valid 8 character standard time. The conversion utility builds the VER(variable,STDTIME) statement.

IPADDR4
Specifies that the variable must contain a valid 15-position IP address. The conversion utility builds the VER(variable,IPADDR4) statement.

Compatibility Considerations

In ISPF Version 3.1, the conversion utility also supported the CHECKI attribute and value TYPE=NUMERIC. If used, the conversion utility will generate a VER(variable ENUM) and a warning message. This support provides backward compatibility with ISPF Version 3.1.

You should now use the TYPE=NUMERIC attribute of the VARCLASS tag to specify that checking for a numeric value should be performed. See "VARCLASS (Variable Class)" on page 482 for more information.

Description

The CHECKI tag defines a test of an input value. Validity checking occurs only on input.

Conditions

- You must code the CHECKI tag within a CHECKL definition. The conversion utility supports only one CHECKI within each CHECKL definition. If multiple CHECKI definitions are coded within a single CHECKL definition, the additional CHECKI tags are ignored and are not syntax checked. See "CHECKL (Validity Check List)" on page 253 for a complete description of this tag.
- The conversion utility builds a VER statement in the ISPF)PROC section of the panel definition for a CHECKI tag.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested Tags

None.

Example

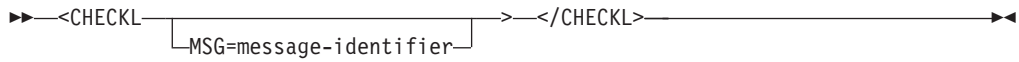
In this example, variables associated with the variable class *aa* must have a value that contains only alphabetic characters. Values associated with the variable class *bb* can only be SINGLE or DOUBLE.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=aa TYPE='char 18'>
  <CHECKL MSG=msgf881>
    <CHECKI TYPE=ALPHA>
  </CHECKL>

<VARCLASS NAME=bb TYPE='char 6'>
  <XLATL FORMAT=upper>
  </XLATL>
  <CHECKL MSG=msgf883>
    <CHECKI TYPE=VALUES PARM1=EQ PARM2="SINGLE DOUBLE">
```

```
  </CHECKL>

<VARLIST>
  <VARDCL NAME=checka VARCLASS=aa>
  <VARDCL NAME=checkb VARCLASS=bb>
</VARLIST>

<PANEL NAME=checki>CHECKI audits
  <DTAFLD DATAVAR=checka ENTWIDTH=18 PMTWIDTH=20>Enter Last Name
  <DTAFLD DATAVAR=checkb ENTWIDTH=6 PMTWIDTH=20>Enter Room Type
  <CMDAREA>
</PANEL>
```

CHECKL (Validity Check List)

The CHECKL tag defines a validity check for input variables.



MSG=message-identifier
This attribute identifies the message to be issued if the value fails the embedded test. The conversion utility adds this *message-identifier* to the VER statement generated by the enclosed CHECKI tag. If you do not specify your own message, ISPF issues a message specified on the enclosing VARCLASS tag or the default message associated with the type of VER statement generated. See "MSG (Message)" on page 393 for information on creating messages.

Description

The CHECKL tag defines a validity check for input variables. The CHECKI tag coded within the check list performs the validation test.

Field validity checking follows standard ISPF conventions based on the verification statements generated. For details, see "CHECKI (Validity Check Item)" on page 244.

Conditions

- The CHECKL tag requires an end tag.
- You must code the CHECKL tag within a VARCLASS definition. See "VARCLASS (Variable Class)" on page 482 for a complete description of this tag.
- The CHECKL tag must be coded after any and all XLATL tags in the same variable class.

Nested Tags

You can code the following tag within a CHECKL definition:

Tag	Name	Usage	Page	Required
CHECKI	Validity check item	Single	244	No

Example

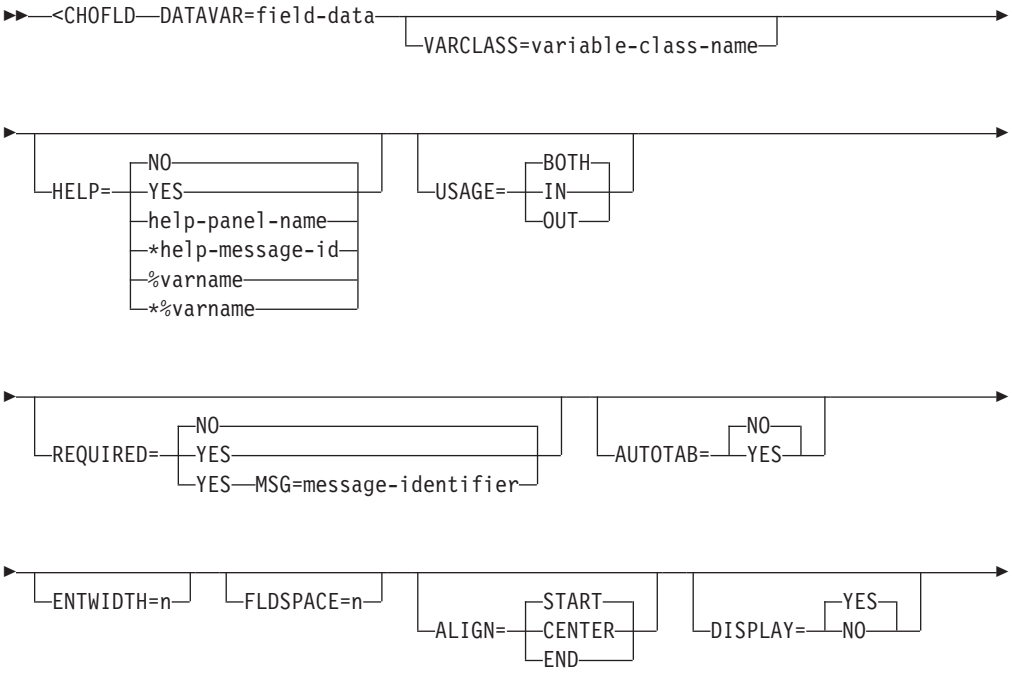
The following source file markup contains two variable classes that each contain a validity check list. The second variable class also contains a translate list that

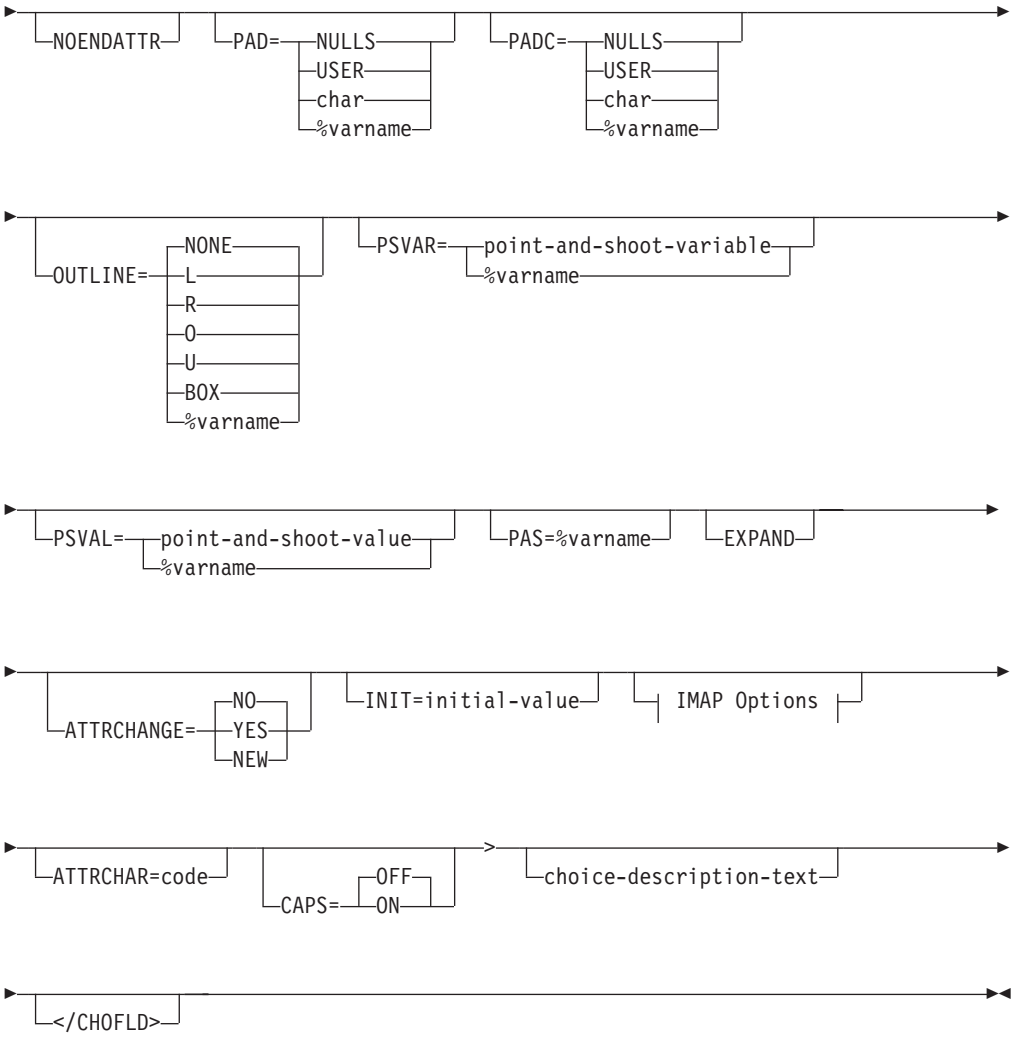
translates variable values to uppercase. Notice that the translate list is coded in the variable class before the validity check list.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=aa TYPE='char 18'>
  <CHECKL MSG=msgf881>
    <CHECKI TYPE=ALPHA>
  </CHECKL>

<VARCLASS NAME=bb TYPE='char 6'>
  <XLATL FORMAT=upper>
  </XLATL>
  <CHECKL MSG=msgf883>
    <CHECKI TYPE=VALUES PARM1=EQ PARM2="SINGLE DOUBLE">
  </CHECKL>

<VARLIST>
  <VARDCL NAME=checka VARCLASS=aa>
  <VARDCL NAME=checkb VARCLASS=bb>
</VARLIST>

<PANEL NAME=checkl>CHECKL audits
  <DTAFLD DATAVAR=checka ENTWIDTH=18 PMTWIDTH=20>Enter Last Name
  <DTAFLD DATAVAR=checkb ENTWIDTH=6 PMTWIDTH=20>Enter Room Type
  <CMDAREA>
</PANEL>
```

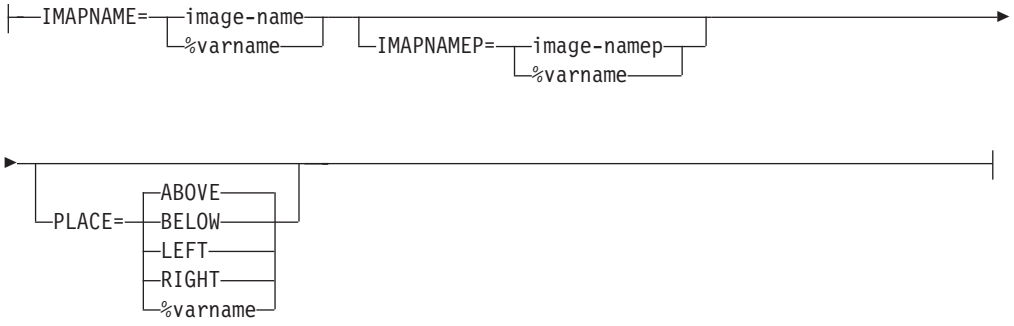
CHOFLD (Choice Data Field)

The CHOFLD tag defines an input field, an output field, or an input/output field within the text of a CHOICE tag.





IMAP Options:



DATAVAR=field-data
This attribute specifies the variable name for the data in the field. The value coded must be a variable-name without the leading % notation.

VARCLASS=variable-class-name
This attribute specifies the name of the variable class, defined using a

VARCLASS tag, that overrides the default variable class referred to by the VARDCL that declared the data variable for this field.

HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname
This attribute specifies the help action taken when the user requests help for this choice data field. This is field-level help.

When HELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help for the choice data field and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

USAGE=BOTH | IN | OUT
This attribute indicates whether the field is for input only, output only, or both.

REQUIRED=NO | YES
This attribute indicates if the field requires input. This attribute is valid only when USAGE=IN or BOTH.

If REQUIRED=YES is coded, a VER(variable,NONBLANK) statement will be built by the conversion utility and placed in the)PROC section of the ISPF panel generated.

MSG=message-identifier
This attribute specifies the message that is displayed when the user does not complete a required entry (defined with the REQUIRED attribute). If you do not specify a *message-identifier*, ISPF displays a default message.

If you specify the MSG attribute and REQUIRED=YES, a VER(variable,NONBLANK,MSG=message-identifier) statement is built by the conversion utility and placed in the)PROC section of the ISPF panel generated. If you specify the MSG attribute and REQUIRED=NO (the default), the conversion utility issues a warning message.

See "MSG (Message)" on page 393 for information on creating messages.

Note: You can specify messages pertaining to other validations using XLATL and CHECKL tags within a VARCLASS definition. See the descriptions of these tags for additional information.

AUTOTAB=NO | YES
When AUTOTAB=YES, the cursor moves to the next field capable of input when the user enters the last character in this field. If no other field capable of user input exists on the panel, the cursor returns to the beginning of this field. The ISPF SKIP keyword is not supported when running in GUI mode.

AUTOTAB=YES is valid only when the value for USAGE is either BOTH or IN. If specified, this attribute overrides the AUTOTAB value of the DTACOL tag.

ENTWIDTH=n
This attribute specifies the number of bytes used for the choice data field. The minimum width is 1 and the maximum is the remaining available panel width, less the required amount of space for field attributes. If ENTWIDTH is not provided on either the CHOFLD tag or the enclosing DTACOL tag, the conversion utility will use the width determined by the TYPE value of the associated VARCLASS.

If specified, this attribute overrides the ENTWIDTH value of the DTACOL tag.

FLDSPACE=n
This attribute specifies the number of bytes reserved for the choice data field. The minimum width is 2 and the maximum is the remaining available panel (or region) width. The FLDSPACE value should include the actual entry width plus the number of entry field attributes. If the value specified by ENTWIDTH is less than the specified FLDSPACE value, the entry field is padded with blanks to the FLDSPACE value. This will create blank space between the entry field and following *choice-description-text* and allows you to align description text from successive CHOFLD definitions.

If specified, this attribute overrides the FLDSPACE value of the DTACOL tag.

ALIGN=START | CENTER | END
This attribute specifies the alignment of data within the display field after all translations have been performed. Use this attribute to align the data with the start, the end, or the center of the display field.

This is accomplished in the conversion utility by using an attribute character for the field that specifies JUST(LEFT) if ALIGN=START or JUST(RIGHT) if ALIGN=END. ALIGN=CENTER will use an attribute character for the field that specifies JUST(ASIS).

Alignment occurs if the transformed value of the dialog variable is shorter than the display width of the field. When ALIGN=END, there is no underscore padding performed. Instead, blanks are used.

DISPLAY=YES | NO
This attribute specifies whether data will display on the screen when the user types it in. If you specify NO, the data will not display. This attribute is useful when creating fields for passwords or other information which you do not want to appear on the screen.

NOENDATTR
This attribute, which is valid only when WINDOW=NO is specified on the PANEL tag or DIR=HORIZ is specified on the REGION tag, specifies that no ending attribute will be placed after the choice data field. NOENDATTR is ignored for the last field on each panel line unless WINDOW=NO has been specified on the PANEL tag. NOENDATTR is valid only when the CHOFLD tag is followed by a CHOFLD, CHOICE, or CHDIV tag.

PAD=NULLS | USER | char | %varname
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

If specified, this attribute overrides the PAD value of the DTACOL tag.

PADC=NULLS | USER | char | %varname
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

If specified, this attribute overrides the PADC value of the DTACOL tag.

OUTLINE=NONE | L | R | O | U | BOX | %varname
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

If specified, this attribute overrides the OUTLINE value of the DTACOL tag.

PSVAR=point-and-shoot-variable | %varname
This attribute provides the name of a variable that is to be set when a CHOFLD is clicked on for point-and-shoot selection. You can define this attribute as a variable name preceded by a "%".

The *point-and-shoot-variable* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

PSVAL=point-and-shoot-value | %varname
This attribute provides the value to be placed in the field specified by the PSVAR attribute. You can define this attribute as a variable name preceded by a "%". To specify a blank value, the "' '" (quotation mark, apostrophe, blank, apostrophe, quotation mark) coding notation should be used.

PAS=%varname
This attribute can be used to provide a variable name to specify ON or OFF for point-and-shoot. When PSVAR and PSVAL have been specified without the PAS attribute, the point-and-shoot field will be automatically enabled.

EXPAND
The EXPAND attribute, used in combination with EXPAND=xy on the PANEL definition, causes the expand characters to be added to the CHOFLD definition, effectively allowing ENTWIDTH to expand. The ENTWIDTH value must be specified as 4 or greater for the EXPAND function to operate.

Note: If the PANEL tag attribute EXPAND is defined with no value specified, the CHOFLD tag EXPAND attribute is not used.

ATTRCHANGE=NO | YES | NEW
When ATTRCHANGE=YES or ATTRCHANGE=NEW, the conversion utility formats an additional entry in the panel)ATTR section (that can apply to multiple data fields) instead of creating a unique .ATTR(field-name) entry in the)INIT section for this field. With this option, multiple CHOFLD tags with the same characteristics require fewer panel logic statements. ATTRCHANGE=NEW creates a new entry. ATTRCHANGE=YES uses an existing entry, if possible.

INIT=initial-value
When the INIT attribute is specified, the conversion utility adds a statement to the panel)INIT section to initialize the field to the *initial-value*.

IMAPNAME= image-name | %varname
This attribute specifies the name of an image to be placed on the point-and-shoot push button when it is displayed in GUI mode. The *image-name* is not used when the panel is displayed in host mode.

The *image-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

IMAPNAMEP=image-namep | %varname
This attribute specifies the name of an image to be placed on the point-and-shoot push button after it has been pushed when it is displayed in GUI mode. The *image-namep* is not used when the panel is displayed in host mode.

The *image-namep* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

PLACE=ABOVE | BELOW | LEFT | RIGHT
This attribute specifies the position of the image relative to the text within the point-and-shoot push button.

ATTRCHAR=code
This attribute can be a single character or a two-position entry of valid hex digits. If you enter an incorrect value, a warning message is issued and the value is set to null. Hex entries are converted to character. Hex values '00'-'2F' are reserved for use by the conversion utility.

CAPS=OFF | ON
When CAPS=ON, the data in the field is displayed in uppercase characters.

choice-description-text
This is the text for the choice data field. The *choice-description-text* appears following the field.

Description

The CHOFLD tag is similar to the DTAFLD tag. When the enclosing SELFLD tag is defined within a DTACOL tag, the CHOFLD tag will use default values defined by the DTACOL tag in the same way as the DTAFLD tag.

The CHOFLD tag defines an input field, an output field, or an input/output field within CHOICE tag description text on an application panel.

The formatted width of the field is 2 positions more than the ENTWIDTH value to provide for an attribute byte both before and after the field.

Conditions

- You must code the CHOFLD tag within a CHOICE tag definition. The CHOFLD tag can be placed anywhere within the *choice-description-text*. See "CHOICE (Selection Choice)" on page 261 for a description of this tag.
- The variable name specified in the DATAVAR attribute should have an associated VARDCL definition. See "VARDCL (Variable Declaration)" on page 486 for a complete description of this tag.
- If both PAD and PADC have been specified, PAD is ignored and PADC is used.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the %varname entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested Tags

You can code the following tags within a CHOFLD definition:

Tag	Name	Usage	Page	Required
ACTION	Action	Multiple	217	No
COMMENT	Comment	Multiple	282	No
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SOURCE	Source	Multiple	472	No

Example

The following source file markup contains an application panel that is similar to the example for the CHOICE tag. In this example, the first selection field is modified to illustrate the CHOFLD tag. The first choice includes a panel input/output field named *cardtype* which must be completed when the *new* choice is selected.

Notice that the reference CHOICE source file has been modified to:

- Add a VARCLASS for the *cardtype* field *before* the include file which has both VARCLASS and VARDCL tags.
- Add a VARDCL for the *cardtype* field *after* the include file which has both VARCLASS and VARDCL tags.
- Add the CHOFLD tag to define the choice data field.
- Add a DTACOL tag definition to allow for a SOURCE tag that provides an audit of *cardtype* only when *new* is selected.

Figure 102 on page 261 shows the formatted result.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampabc system>)>

<VARCLASS NAME=char9cls TYPE='char 9'>
  <XLATL FORMAT=upper>
  </XLATL>

&sampvar1;

<VARLIST>
  <VARDCL NAME=cardtype VARCLASS=char9cls>
</VARLIST

<PANEL NAME=chofld KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
<TOPINST>Type in patron's name and card number (if applicable).
<TOPINST>Then select an action bar choice.
<AREA>
  <DTAFLD DATAVAR=curdate PMTWIDTH=12 ENTWIDTH=8 USAGE=out>Date
  <DTAFLD DATAVAR=cardno PMTWIDTH=12 ENTWIDTH=7 DESWIDTH=25>Card No
    <DTAFLDD>(A 7-digit number)
  <DTAFLD DATAVAR=name PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25>Name
    <DTAFLDD>(Last, First, M.I.)
  <DTAFLD DATAVAR=address PMTWIDTH=12 ENTWIDTH=25>Address
  <DIVIDER>
  <REGION DIR=horiz>
  <SELFLD NAME=cardsel PMTWIDTH=30 SELWIDTH=38>Choose
  one of the following
    <CHOICE CHECKVAR=card MATCH=new>
            New    Type:
    <CHOFLD datavar=cardtype autotab=yes entwidth=9>
          (Permanent or Temporary)
<CHOICE CHECKVAR=card MATCH=renew>Renewal
<CHOICE CHECKVAR=card MATCH=replace>Replacement
  </SELFLD>
 <DTACOL>
    <SOURCE>
IF (&CARDSEL = 1)
  VER(&CARDTYPE,NB,LIST,TEMPORARY,PERMANENT)
    </SOURCE>
  </DTACOL>
  <SELFLD TYPE=multi PMTWIDTH=30 SELWIDTH=25>Check valid branches
```

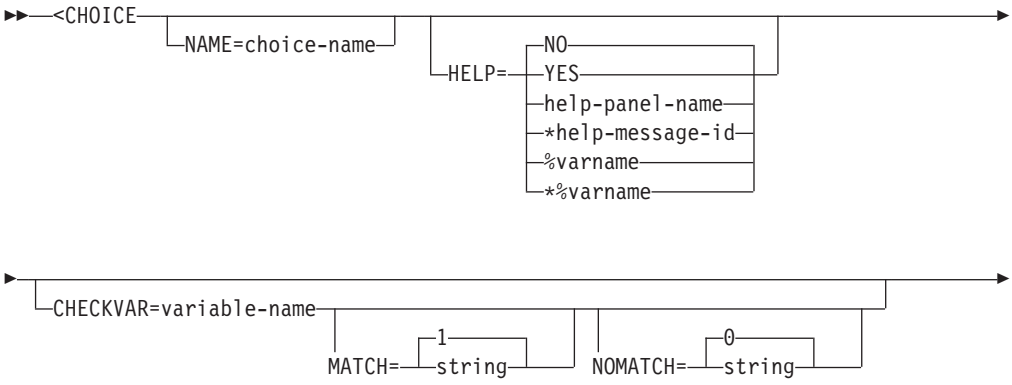
```
   <CHOICE NAME=north HELP=nthhlp CHECKVAR=nth>North Branch
   <CHOICE NAME=south HELP=sthhlp CHECKVAR=sth>South Branch
   <CHOICE NAME=east HELP=esthlp CHECKVAR=est>East Branch
    <CHOICE NAME=west HELP=wsthlp CHECKVAR=wst>West Branch
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

```
  File  Search  Help
--------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number (if applicable).

Then select an action bar choice.

Date . . . :
Card No. . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following           Check valid branches
__  1.  New      Type: Z             _  North Branch
        (Permanent or Temporary)     _  South Branch
    2.  Renewal                      _  East Branch
    3.  Replacement                  _  West Branch

Enter a command ===> ____________________________________________________________
 F1=Help        F2=Split       F3=Exit        F6=KEYSHELP    F9=Swap
F12=Cancel
```

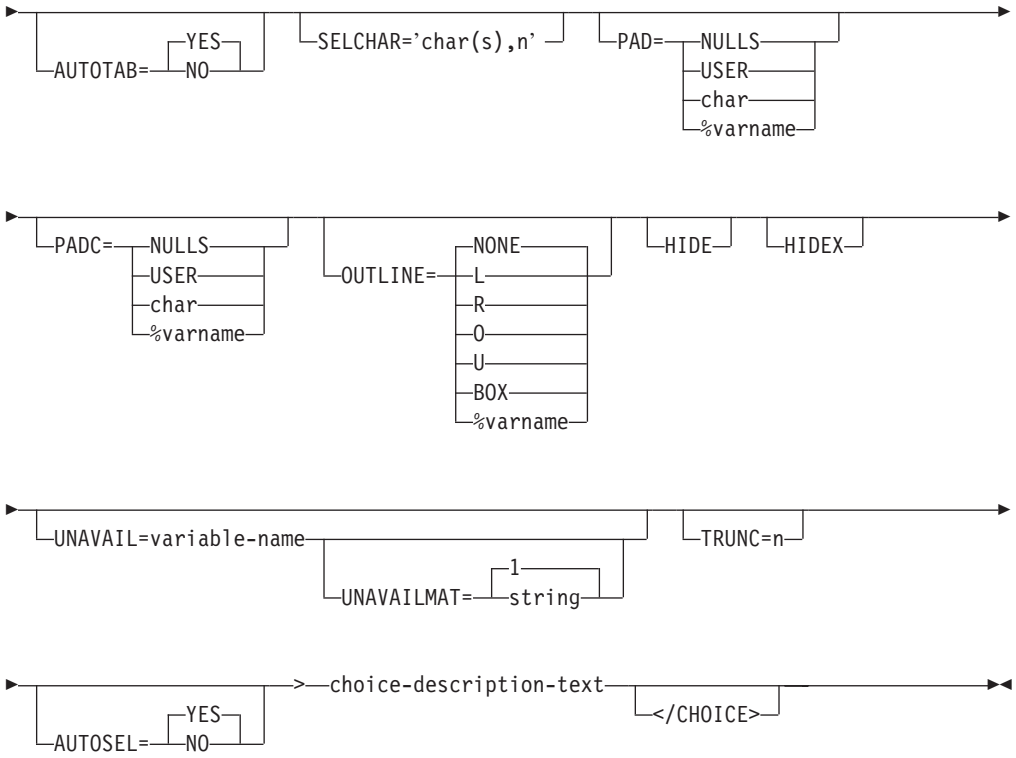
Figure 102. Choice Data Fields

CHOICE (Selection Choice)

The CHOICE tag defines information about a choice in a selection field.



CHOICE



NAME=choice-name
Specifies the name of the choice. The *choice-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

Note: This attribute is required for choices defined for a multiple-choice selection field because the *choice-name* is used as the input field for multiple choice selections.

For multiple-choice selection fields, the *choice-name* can also be used to position the cursor on the choice or to position a pop-up.

Note: This attribute is not supported by the conversion utility for single-choice selection fields. In this case, the NAME value of the SELFLD tag is used as the field name.

HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname
This attribute specifies the help action taken when the user requests for a multiple-choice selection field. This is field-level help.

When HELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help on a choice and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=MULTI.

CHECKVAR=variable-name
This attribute defines a variable whose value indicates whether or not the choice is preselected when the selection field is displayed. If the value of the variable is equivalent to the *string* you specify with the MATCH attribute, the item is marked as selected when the panel displays.

The preselection indicator depends on the value of the TYPE attribute from the SELFLD tag and whether the display mode is host or GUI.

TYPE	LISTTYPE	Host Display Indicator	GUI Display Indicator
MULTI	n/a	slash	check
SINGLE	(not used) RADIO LISTBOX DDLIST COMBO	Choice number Choice number Choice number Choice number Choice number	Choice number Radio button selected Choice highlighted in list Choice displayed in field Choice placed in field
MENU	n/a	Choice number	Choice number
MODEL	n/a	Choice number	Choice number
TUTOR	n/a	Choice number	Choice number

When the SELFLD tag has been specified with TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR, the CHOICE number (or SELCHAR value) is placed in the command line.

The *variable-name* is updated to the value specified by the MATCH attribute when the user selects the choice being defined. For multiple-choice selection fields (SELFLD TYPE=MULTI), if you do not select a choice, or you deselect a choice, the associated *variable-name* is set to the value of the NOMATCH attribute or to 0 if the NOMATCH attribute is not specified.

Use a different variable for *variable-name* than what has been specified for *choice-name*.

Do not use the same variable for the *variable-name* as you use for the *variable-name* specified for the SETVAR or TOGVAR attributes of the ACTION tag.

For single-choice selection fields (SELFLD TYPE=SINGLE), ISPF selection menus (SELFLD TYPE=MENU), edit model selection menus (SELFLD TYPE=MODEL), or tutorial selection menus (SELFLD TYPE=TUTOR), the *variable-name* should be the same for all of the choices. For multiple-choice selection fields (SELFLD TYPE=MULTI), the *variable-name* should be different for each choice.

The CHECKVAR attribute value must be specified without a leading % sign. The *variable-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

MATCH=1 | string
Defines the value for the check variable that causes the item to be preselected. The *string* can be any character string. MATCH=1 is the default.

NOMATCH=0 | string
Defines the value for setting the check variable when the item is not selected. NOMATCH=0 is the default.

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=MULTI.

AUTOTAB=YES | NO
When AUTOTAB=YES, the cursor moves to the next field capable of input when the user enters the last character in this field. If no other field capable of user input exists on the panel, the cursor remains on this field.

The ISPF SKIP keyword is not supported when running in GUI mode.

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=MULTI.

SELCHAR='char(s),n'
This attribute specifies an alphanumeric character(s) to be used as the selection menu, edit model selection menu, or tutorial selection menu choice in place of the normal numeric value automatically supplied by the conversion utility. The number of characters accepted is controlled by the ENTWIDTH attribute of the SELFLD tag. The *char(s)* value is used as coded, that is, it is not uppercase.

When the HIDE attribute is also specified, the number of characters to be used for the hidden choice selection may be specified as part of the SELCHAR attribute. If specified, the *n* value overrides the number of characters normally obtained from the ENTWIDTH attribute of the SELFLD tag. The *n* value can be a numeric value from 1 to the number of bytes provided as the *char(s)* value, or you can specify an "*" to tell the conversion utility to use all of the *char(s)* provided for the choice selection. The *n* value is ignored when the HIDE attribute is not specified.

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR.

PAD=NULLS | USER | char | %varname
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=MULTI.

PADC=NULLS | USER | char | %varname
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=MULTI.

OUTLINE=NONE | L | R | O | U | BOX | %varname
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=MULTI.

HIDE
This attribute causes a choice entry for a single-choice, menu-choice, model-choice, or tutor-choice selection to be removed from the selection list display.

This allows the creation of a numbered selection list when the choice numbers are not continuous by adding a 'dummy' CHOICE tag at the appropriate place in the DTL source. The number assigned to the hidden CHOICE does not appear in the selection list. Normal)INIT and)PROC section entries are not affected.

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=SINGLE, TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR.

HIDEX
This attribute causes a choice entry for a model-choice selection to be removed both from the selection list display and from the selection processing.

This attribute is used in combination with the TRUNC attribute and the SELCHAR attribute to supply an alternate CHOICE tag definition with an alternate hidden model selection keyword.

For example, if an edit model panel has a selectable description of "VER", but you also want to allow the full word "VERIFY" to select the same model, two CHOICE tags are required. The first one defines the choice with the text "VER". The alternate CHOICE uses the same SELCHAR information, adds the attribute HIDEX and TRUNC=3, and specifies the tag text as "VERIFY". The conversion utility uses the first definition to build the panel text and the selection processing statement and uses the alternate CHOICE to accept the entry "VERIFY" by truncating it to "VER".

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=MODEL.

UNAVAIL=variable-name
This attribute defines a variable whose value indicates whether or not the choice is available when the selection field is displayed. If the value of the variable is equivalent to the *string* you specify with the UNAVAILMAT attribute (or to the default value "1"), the item is displayed as an unavailable choice.

UNAVAILMAT=1 | string
Defines the value for the UNAVAIL variable that causes the choice to be unavailable. The *string* can be any character string. UNAVAILMAT=1 is the default.

TRUNC=n
This attribute is used for model-choice selection to specify the minimum number of characters required to identify the model choice. If the TRUNC attribute is not specified, the entire model choice name must be used to identify the model selection.

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=MODEL.

AUTOSEL=YES | NO
This attribute is used for tutor-choice selection to control the automatic selection of this choice by tutorial processing. When AUTOSEL=NO, the choice is not automatically selected.

choice-description-text
The text of the selection choice.

Description

The CHOICE tag defines a choice within a selection field. The behavior and appearance of the choice depends on whether it is coded within a single-choice, multiple-choice, or menu-choice selection field. The single-choice entries are further affected in GUI mode by the value of the LISTTYPE attribute on the SELFLD tag.

For menu-choice selection fields, the text is preceded by a number (not followed by a period), the input field is the command line, and the choice selection is displayed with the CUA type Normal Text (NT).

For a single-choice selection list:
- When the LISTTYPE attribute of the SELFLD tag is not specified, the text is preceded by a number (followed by a period), the conversion utility provides an input field before the first choice for entry of the number of the selected choice, and the choice selection is displayed with the CUA type Select Available Choices (SAC).
- When LISTTYPE=RADIO is specified on the SELFLD tag, the choice selection is displayed as a radio button in GUI mode.
- When LISTTYPE=LISTBOX is specified on the SELFLD tag, the choice selection is displayed as a list box in GUI mode.
- When LISTTYPE=DDLIST is specified on the SELFLD tag, the choice selection is displayed as a drop-down list in GUI mode.
- When LISTTYPE=COMBO is specified on the SELFLD tag, the choice selection is displayed in a combination box in GUI mode.

The field name for single-choice selection fields is the value specified for the NAME attribute of the SELFLD tag. The default field name for an ISPF selection menu choice is the field name used to identify the command line, normally ZCMD.

The text of each choice in a multiple-choice selection field is preceded by an input field. The field name for multiple-choice selection fields is the value specified for the NAME attribute of the CHOICE tag.

You can define an action for each choice using the SETVAR or TOGVAR attribute in an ACTION tag associated with the choice. Typically, an application knows what choice was selected by the application user by the value in the selection field name. The CHOICE field name for a multi-choice selection is set to a "/" when control is returned to the application. The SELFLD field name contains the number of the choice for single choice selection when control is returned to the application. The command line variable name contains the number of a menu selection choice when control is returned to the application. Alternatively, the application can use the value of the check variable or use SETVAR or TOGVAR to set another named variable.

Conditions

- You must code the CHOICE tag within a SELFLD definition. See "SELFLD (Selection Field)" on page 455 for a complete description of this tag.
- If coded within a multiple-choice selection field (SELFLD TYPE=MULTI), the *choice-name* can have an associated VARDCL definition.
- If both PAD and PADC have been specified, PAD is ignored and PADC is used.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.
- If the *choice-description-text* contains HP (Emphasized Text) or RP (Reference Phrase) tags, the UNAVAIL attribute is ignored.

Nested Tags

You can code the following tags within a CHOICE definition:

Tag	Name	Usage	Page	Required
ACTION	Action	Multiple	217	No
CHOFLD	Choice data field	Multiple	254	No
COMMENT	Comment	Multiple	282	No
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SOURCE	Source	Multiple	472	No

Example

The application panel in the following markup contains two selection fields. The first is a single-choice selection field that can be preselected depending on the value assigned to the variable *card*. When *card* is equal to *new*, *renew*, or *replace*, the selection field's input data field is assigned a value of 1, 2, or 3, respectively; otherwise, it is not preselected and the input data field remains blank.

The second selection field is a multiple-choice selection field. This field can be preselected by assigning values to the variables *nth*, *sth*, *est* and *wst*. If the given variable equals 1, the corresponding selection field is marked with a /. More than one of the choices may be selected. Any non-blank character in the choice entry-field will select that choice. Preselected choices can be deselected by typing a blank character over the field.

Figure 103 on page 268 shows the formatted result.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampabc system>)>
&sampvar1;

<PANEL NAME=choice1 KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
<TOPINST>Type in patron's name and card number (if applicable).
<TOPINST>Then select an action bar choice.
<AREA>
  <DTAFLD DATAVAR=curdate PMTWIDTH=12 ENTWIDTH=8 USAGE=out>Date
```

```
  <DTAFLD DATAVAR=cardno PMTWIDTH=12 ENTWIDTH=7 DESWIDTH=25>Card No
    <DTAFLDD>(A 7-digit number)
  <DTAFLD DATAVAR=name PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25>Name
    <DTAFLDD>(Last, First, M.I.)
  <DTAFLD DATAVAR=address PMTWIDTH=12 ENTWIDTH=25>Address
  <DIVIDER>
  <REGION DIR=horiz>
  <SELFLD NAME=cardsel PMTWIDTH=30 SELWIDTH=38>Choose
  one of the following
    <CHOICE CHECKVAR=card MATCH=new>New
    <CHOICE CHECKVAR=card MATCH=renew>Renewal
    <CHOICE CHECKVAR=card MATCH=replace>Replacement
  </SELFLD>
  <SELFLD TYPE=multi PMTWIDTH=30 SELWIDTH=25>Check valid branches
    <CHOICE NAME=north HELP=nthhlp CHECKVAR=nth>North Branch
    <CHOICE NAME=south HELP=sthhlp CHECKVAR=sth>South Branch
    <CHOICE NAME=east HELP=esthlp CHECKVAR=est>East Branch
    <CHOICE NAME=west HELP=wsthlp CHECKVAR=wst>West Branch
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

```
  File  Search  Help
------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number (if applicable).

Then select an action bar choice.

Date . . . :
Card No. . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following          Check valid branches
__  1.  New                         _  North Branch
    2.  Renewal                     _  South Branch
    3.  Replacement                 _  East Branch
                                    _  West Branch

Enter a command ===> ___________________________________________________________
 F1=Help        F2=Split       F3=Exit        F6=KEYSHELP    F9=Swap
F12=Cancel
```

Figure 103. Selection Field Choices

CMD (Command Definition)

The CMD tag defines a command within an application command table.

NAME=internal-command-name
This attribute specifies an internal name for the command. The *internal-command-name* must have the following characteristics:
- 2–8 single-byte characters in length
- The first (or only) character must be A–Z, a–z, @, #, or $.
- Remaining characters, if any, can be A–Z, a–z, @, #, $, —, or 0–9.

Lowercase characters are translated to their uppercase equivalents.

The *internal-command-name* is used in two ways:
- As the command table search criteria when:
  - A key defined in the current key list is pressed
  - A pull-down choice with an associated RUN action is selected
  - A command is entered in the command area of a panel.
- As the value passed to dialogs when the command action is PASSTHRU or SETVERB. See "CMDACT (Command Action)" on page 270 for more information on the PASSTHRU and SETVERB command actions.

ALTDESCR=command-description
This attribute provides a description of the command. It is placed in the ISPF variable ZCTDESC. The *command-description* text length is limited to 80 bytes.

external-command-name
Specifies the external name for this command.

Note: The *external-command-name* must be equal to the *internal-command-name*. You must use the *external-command-name* to support the ability provided by ISPF for truncated command entry and the T (truncation) tag. For more information, see "T (Truncation)" on page 474.

Description

The CMD tag defines a command within an application command table. The defined command can be issued by an application user by entering the *internal-command-name* in the panel command area, or pressing a function key, or selecting a pull-down choice that references the command's *internal-command-name.* See "KEYI (Key Item)" on page 358 and "ACTION (Action)" on page 217 for additional information.

The action to be taken when a command is issued is defined with the CMDACT tag. See "CMDACT (Command Action)" on page 270 for information on defining command actions.

Conditions

- The CMD tag must be coded within a CMDTBL definition. See "CMDTBL (Command Table)" on page 280 for a complete description of this tag.

Nested Tags

You can code the following tags within a CMD definition:

Tag	Name	Usage	Page	Required
CMDACT	Command action	Single	270	Yes
T	Truncation	Single	474	No

Example

The following source file markup contains a command table that defines the commands UPDATE, ADD, DELETE, and SEARCH. The DELETE and UPDATE commands have defined truncations.

```
<!DOCTYPE DM SYSTEM>

<CMDTBL APPLID=conv>
  <CMD NAME=update>Upd<T>ate
    <CMDACT ACTION='alias add'>
  <CMD NAME=add>Add
    <CMDACT ACTION=setverb>
  <CMD NAME=delete>Del<T>ete
    <CMDACT ACTION=passthru>
  <CMD NAME=search>Search
    <CMDACT ACTION=passthru>
</CMDTBL>
```

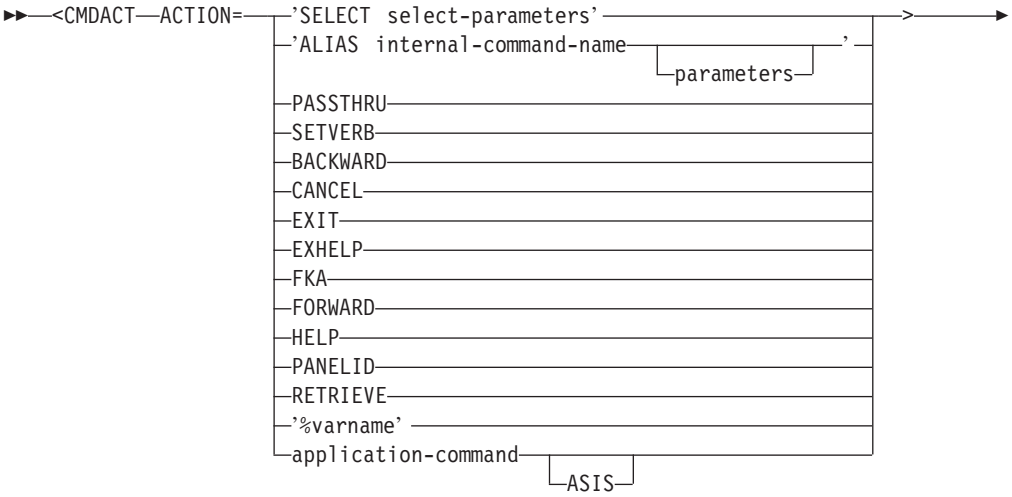
The following table shows the resultant ISPF application command table.

Table 1. ISPF Application Command Table

ZCTVERB	ZCTTRUNC	ZCTACT
UPDATE	3	ALIAS ADD
ADD	0	SETVERB
DELETE	3	PASSTHRU
SEARCH	0	PASSTHRU

CMDACT (Command Action)

The CMDACT tag defines the action that occurs when the associated command is issued.



ACTION=

This attribute indicates the action that should be performed when the associated command is issued. The ACTION attribute value is limited to 240 characters. The value must be one of the following:

SELECT select-parameters
Causes the ISPF SELECT service to be issued.

ALIAS internal-command-name
Provides an alternate way to express a command. For example, you can assign QUIT as an alias for the command EXIT.

The ALIAS *internal-command-name* has a maximum length of 8 characters.

In the command table, an alias must precede the command for which it is an alias.

You can create a chain of command aliases in a command table, as long as the result is a valid executable action. The last command and parameter values that ISPF encounters in the alias chain are the ones executed. The command and the parameter values do not necessarily come from the same command definition entry. For example:

Command Name	Command Action
EASYKEY	ALIAS CMD PARM1 PARM2
CMD	ALIAS CMD1 PARM3
CMD1	ALIAS CMD2

In this example, if the EASYKEY command is issued, the command that would ultimately be executed would be CMD2 PARM3.

parameters
If any ALIAS parameters are specified, they take precedence over any parameters included with the command when issued from a command line or the ACTION tag RUN attribute when a pull-down choice is selected.

If the ALIAS *internal-command-name* does not include parameters, ISPF accepts parameters from the command line or ACTION tag.

PASSTHRU
The PASSTHRU action causes the command and any parameters to be passed to the dialog program in the ZCMD dialog variable.

SETVERB
This is an alternate way to pass a command to the dialog. The SETVERB action causes the *internal-command-name* to be passed to the dialog in the ZVERB dialog variable. Any command parameters are passed in the ZCMD dialog variable.

BACKWARD
Specifies the ISPF system command BACKWARD as the command action.

CANCEL
Specifies the ISPF system command CANCEL as the command action.

EXIT
Specifies the ISPF system command EXIT as the command action.

EXHELP
Specifies the ISPF system command EXHELP as the command action.

FKA
Specifies the ISPF system command FKA as the command action.

FORWARD
Specifies the ISPF system command FORWARD as the command action.

HELP
Specifies the ISPF system command HELP as the command action.

PANELID
Specifies the ISPF system command PANELID as the command action.

RETRIEVE
Specifies the ISPF system command RETRIEVE as the command action.

%varname
You can specify a command action dynamically at run time by specifying the name of a variable (using % notation) for the ACTION attribute. If you specify a variable name, ISPF retrieves the action value when the command is issued. The variable value must be one of the actions previously listed.

The "%varname" entry must follow the naming conventions described in "Rules for "%variable" Names" on page 211.

application-command
Specifies an application-unique command as the command action. The command action is created as an ALIAS unless the ASIS keyword is specified.

ASIS Specifies that the application-unique command is to be created without the ALIAS designation.

Description

The CMDACT tag defines the action that occurs when the associated command is issued.

Conditions

- The CMDACT tag must be coded within the CMD definition it is associated with. See "CMD (Command Definition)" on page 268 for a complete description of this tag.
- You must specify the ACTION attribute on the CMDACT tag.

Nested Tags

None.

Example

The following source file markup contains a command table that defines the commands UPDATE, ADD, DELETE and SEARCH. The ADD command sets the ZVERB variable equal to *add*. The DELETE command sets the ZCMD variable to *delete*. The UPDATE command is an alias for ADD.

```
<!DOCTYPE DM SYSTEM>

<CMDTBL APPLID=conv>
  <CMD NAME=update>Upd<T>ate
    <CMDACT ACTION='alias add'>
  <CMD NAME=add>Add
    <CMDACT ACTION=setverb>
  <CMD NAME=delete>Del<T>ete
    <CMDACT ACTION=passthru>
  <CMD NAME=search>Search
    <CMDACT ACTION=passthru>
</CMDTBL>
```

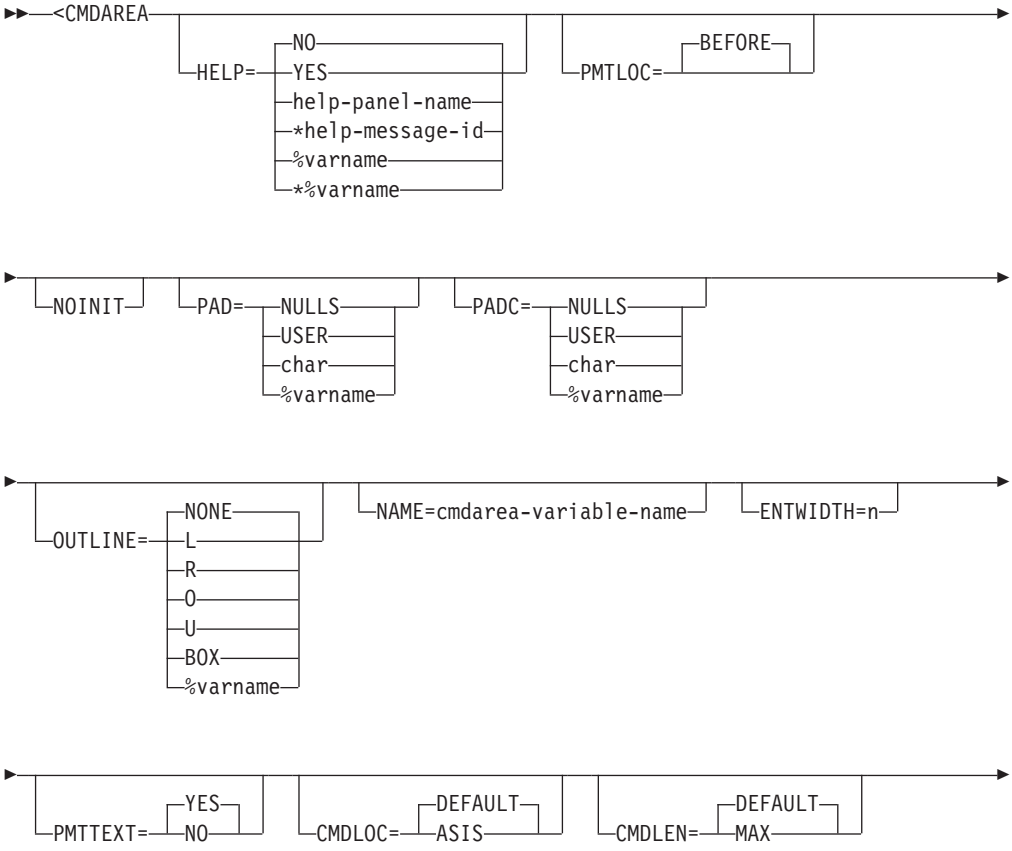
The following table shows the resultant ISPF application command table.

Table 2. ISPF Application Command Table

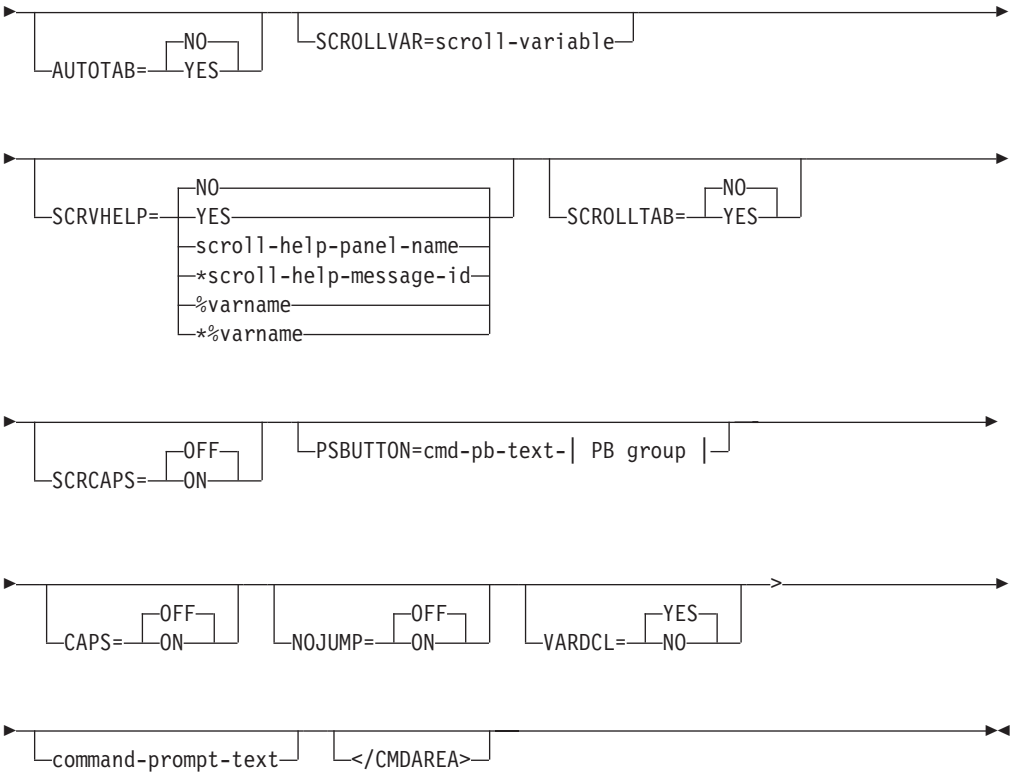
ZCTVERB	ZCTTRUNC	ZCTACT
UPDATE	3	ALIAS ADD
ADD	0	SETVERB
DELETE	3	PASSTHRU
SEARCH	0	PASSTHRU

CMDAREA (Command Area)

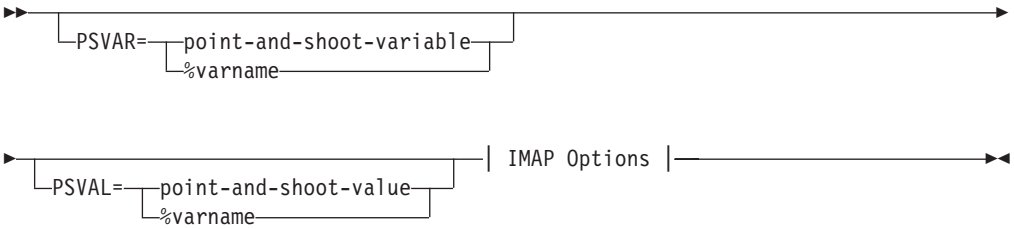
The CMDAREA tag defines a command entry area on an application panel.



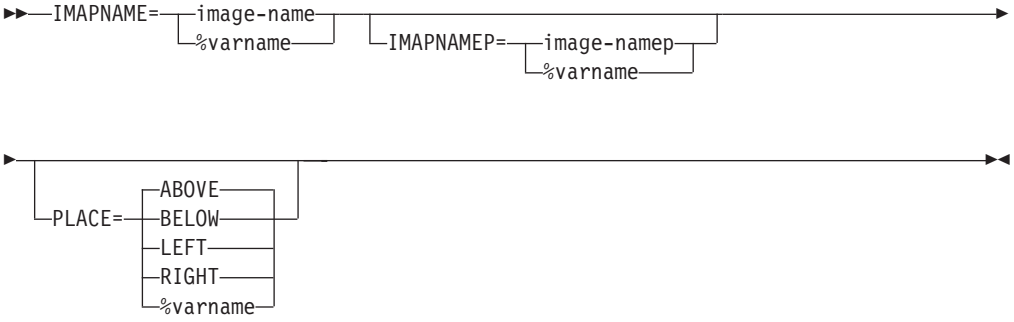
CMDAREA



PB Group:



IMAP Options



HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname
This attribute specifies the help action taken when the user requests help for the command area.

When HELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help on a choice and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

PMTLOC=BEFORE
This attribute defines the location of the prompt text. The text defined by *command-prompt-text* appears on the same line as the command area entry field.

NOINIT
This attribute controls the initial display of the command line. When this attribute is specified, the ZCMD field is not initialized to blanks before the panel is displayed.

PAD=NULLS | USER | char | %varname
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

PADC= NULLS | USER | char | %varname
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

OUTLINE=NONE | L | R | O | U | BOX | %varname
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

NAME=cmdarea-variable-name
This attribute specifies a command area name to replace the default name ZCMD.

The *cmdarea-variable-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

ENTWIDTH=n
This attribute is used to specify the length of the command field. It is used in combination with WINDOW=NO on the PANEL tag to create a command line which is longer than a single panel line.

PMTTEXT=YES | NO
This attribute is used to control the formatting of the *command-prompt-text*. When PMTTEXT=NO, the *command-prompt-text* is not used, leaving only the "===>" indicator for the command field.

CMDLOC=DEFAULT | ASIS
This attribute is used to control the placement of the command line in the generated panel. When CMDLOC=DEFAULT (or when CMDLOC is not specified) the command area is placed at line 2 in the panel, and the display position is controlled by the option specified on the Settings panel. When CMDLOC=ASIS is specified, the command area is placed in the generated panel in the same relative position as the CMDAREA tag is found in the DTL source, and the Settings option is ignored when the panel is displayed.

CMDLEN=DEFAULT | MAX
This attribute is used to control the length of the command line in the generated panel. When CMDLEN=DEFAULT (or when CMDLEN is not specified) the command line length is taken from the specified (or defaulted) WIDTH attribute of the PANEL tag. When CMDLEN=MAX is specified, the command line length is taken from the record length of the output panel file.

This attribute is valid only when WINDOW=NO is specified on the PANEL tag.

AUTOTAB=NO | YES
When AUTOTAB=YES, the cursor moves to the next input field when you enter the last character in the command field. If there is no other input field on the panel, the cursor returns to the beginning of the command line. The ISPF SKIP keyword is not supported in GUI mode.

SCROLLVAR=scoll-variable
This attribute specifies the name of a variable that the application uses to obtain scrolling information. The *scroll-variable* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

If the attribute is specified, the conversion utility creates a scroll entry on the command line, providing that the resulting command area allows at least 8 bytes for a command entry.

SCRVHELP=NO | YES | scroll-help-panel-name | *scroll-help-message-id | %varname | *%varname
This attribute specifies the help action taken when the user requests help for the field specified with the SCROLLVAR attribute.

When SCRVHELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help on a choice and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *scroll-help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

SCROLLTAB=NO | YES
When SCROLLTAB=YES, the cursor moves to the next input field when you enter the last character in the scroll amount field. If there is no other input field on the panel, the cursor returns to the beginning of the command line. The ISPF SKIP keyword is not supported in GUI mode.

SCRCAPS=OFF | ON
When SCRCAPS=ON, the data in the scroll field is displayed in uppercase characters.

PSBUTTON=cmd-pb-text
This attribute requires that the PSVAR and PSVAL attributes also be specified.

This attribute specifies that a command push button is to be placed at the end of the command line, provided that the resulting command area allows at least 8 bytes for a command entry. The push button text area is created as a point-and-shoot field.

PSVAR=point-and-shoot-variable | %varname
This attribute provides the name of a variable that is to be set when the *cmd-pb-text* is clicked on for point-and-shoot selection. You can define this attribute as a variable name preceded by a percent (%) sign.

The *point-and-shoot-variable* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

PSVAL=point-and-shoot-value | %varname
This attribute provides the value to be placed in the field specified by the PSVAR attribute. You can define this attribute as a variable name preceded by a percent (%) sign. To specify a blank value, use the coding notation **"' '"** (quotation mark, apostrophe, blank space, apostrophe, quotation mark).

IMAPNAME=image-name | %varname
This attribute specifies the name of an image to be placed on the point-and-shoot push button when it is displayed in GUI mode. The *image-name* is not used when the panel is displayed in host mode. The *image-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

IMAPNAMEP=image-namep | %varname
This attribute specifies the name of an image to be placed on the point-and-shoot push button after it has been pushed when it is displayed in GUI mode. The *image-namep* is not used when the panel is displayed in host mode. The *image-namep* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

PLACE=ABOVE | BELOW | LEFT | RIGHT | %varname
This attribute specifies the position of the image relative to the text within the point-and-shoot push button.

CAPS=OFF | ON
When CAPS=ON, the data in the field is displayed in uppercase characters.

NOJUMP=OFF | ON
When NOJUMP=ON, the JUMP function is disabled for the field.

VARDCL=YES | NO
When VARDCL=NO the *cmdarea-variable-name* is not checked to the declared variable information provided with the VARCLASS and VARDCL tags.

command-prompt-text
The *command-prompt-text* specifies the prompt text for the command entry area. The maximum prompt text (not including the command area prefix ===>) is 59 bytes for a standard 76 byte-width panel. The conversion utility reserves 8 bytes for a minimum command entry field and 3 additional bytes are required for panel attributes. One blank is placed between the *command-prompt-text* and the command area prefix. One blank is placed between the end of the

command line and the right panel boundary (unless the WINDOW=NO attribute has been specified) to prevent the cursor from skipping into the right panel window border. These formatting considerations mean that the maximum length of the *command-prompt-text* for a panel 76 bytes in width is 59. If the length of the *command-prompt-text* exceeds the available space, a message is issued and the *command-prompt-text* is truncated. If your panel requires that the Scroll field be added to the Command line, or the SCROLLVAR attribute is specified in the CMDAREA definition, the *command-prompt-text* must be further reduced to allow for the Scroll field. If your panel specifies the PSBUTTON attribute, the *command-prompt-text* must be further reduced to allow for the Command push button.

If you do not provide *command-prompt-text*, the word "Command" (or its translated equivalent) is the default, unless you are creating an ISPF selection panel, in which case the word "Option" (or its translated equivalent) is the default. The Common User Access command area prefix (`===>`) is always added automatically in front of the entry field.

Description

The CMDAREA tag defines a command entry area on an application panel. The command entry area extends to the right side of the panel, unless limited by the ENTWIDTH attribute or the presence of a Scroll field. Application users use the command entry area to enter commands.

Note: If you specify the CMDAREA tag within your DTL source file:

- It must appear before the AREA, DA, GA, REGION, or SELFLD tag when DEPTH=* is specified.
- It must appear before the SELFLD tag when TYPE=MENU and CHECKVAR or UNAVAIL attributes are specified on nested CHOICE tags.

Conditions

- You must code the CMDAREA tag within a PANEL definition. You can code only one command area definition for each panel. See "PANEL (Panel)" on page 415 for a complete description of this tag.
- The data entered on the command line is processed "as is". To translate the data to uppercase, you must either provide a VARDCL definition for the field ZCMD with a reference to a VARCLASS containing an XLATL tag which specifies FORMAT=UPPER, or specify CAPS=ON.
- You cannot code the CMDAREA tag within an AREA definition. The Command area is generated at the top of the panel source to allow for floating of the command line. Refer to the *ISPF User's Guide* for more information.
- If both PAD and PADC have been specified, PAD is ignored and PADC is used.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested Tags

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No

Example

The following application panel markup contains a command area. The *command-prompt-text* "Use this area to enter a command" is specified in the markup to override the default text "Command". Figure 104 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>
<VARCLASS NAME=choiccls TYPE='char 2'
<VARCLASS NAME=vccmd TYPE='char 62'>
  <XLATL FORMAT=upper>
  </XLATL>
</VARCLASS>

<VARLIST>
  <VARDCL NAME=sample VARCLASS=choiccls>
  <VARDCL NAME=zcmd  VARCLASS=vccmd>
</VARLIST>

<PANEL NAME=cmdarea1>Choose a Virtue
  <TOPINST>Select a choice.
  <AREA>
    <SELFLD NAME=sample PMTWIDTH=10 SELWIDTH=20>Virtues:
      <CHOICE>Faith
      <CHOICE>Hope
      <CHOICE>Charity
    </SELFLD>
  </AREA>
  <BOTINST>Now press Enter.
 <CMDAREA>Use this area to enter a command
</PANEL>
```

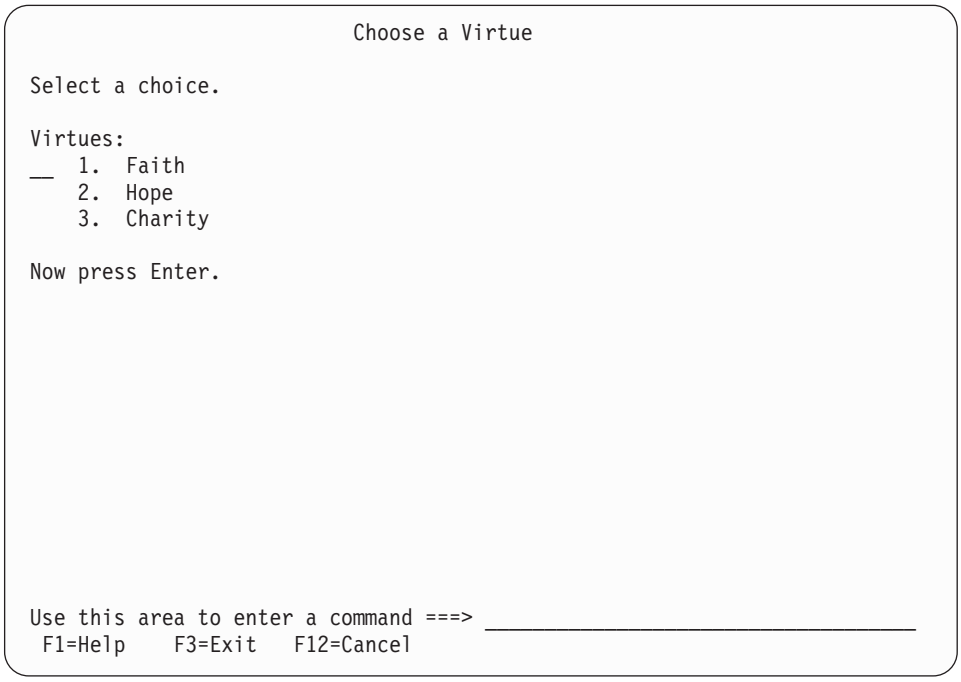
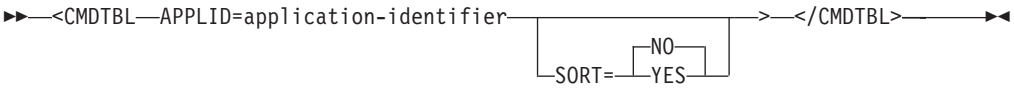


Figure 104. Command Area

CMDTBL (Command Table)

The CMDTBL tag provides support to define the ISPF application command table.



APPLID=application-identifier
This attribute specifies an application identifier. This identifier is used as a prefix to the string "CMDS" to form the name of the command table. The *application-identifier* must have the following characteristics:

- 1–4 characters in length
- The first (or only) character must be A–Z or a–z @, #, or $.
- Remaining characters, if any, must be A–Z, a–z, @, #, $, or 0–9.

Lowercase characters are translated to their uppercase equivalents.

The name of the command table is member name `xxxxCMDS`, where xxxx represents the *application-identifier*.

Command tables are updated using ISPF table services. Input is obtained from the ISPTLIB DDname allocation and output is written to the ISPTABL DDname allocation. Refer to the description of how to allocate libraries before starting ISPF in the *ISPF User's Guide* for more information about the use of ISPTLIB and ISPTABL.

SORT=NO | YES
When SORT=YES is specified, the command table is sorted in command-name sequence. Any commands defined as an ALIAS to other commands are placed in the command table first, in command-name sequence. The regular commands follow the ALIAS entries in command-name sequence.

If SORT=NO or the SORT attribute is not specified, commands are placed in the command table in the sequence the CMD tags are encountered in the DTL source file.

Description

The command table tag provides support to define the ISPF application command table. ACTION tags and definitions of key lists reference the command definitions within an application command table.

Note: To access commands through the use of the key list function keys, specify the KEYLAPPL ID invocation parameter for the conversion utility with the same APPLID value used for the CMDTBL tag.

Note: You can use the TSO ISPCMDTB command to convert existing command tables to DTL. To use ISPCMDTB, ensure that the command table is in your table concatenation (ISPCMDTB), type **TSO ISPCMDTB** *applid* (where applid is the application id of the command table). This will place you in an edit session containing the DTL version of the command table. Use the editor CREATE or REPLACE commands to save the table to your DTL source data set.

Conditions

- The CMDTBL tag requires an end tag.
- You cannot code the CMDTBL tag within any other tag definition.
- You can code only one command table for any application.

Nested Tags

You can code the following tag within a CMDTBL definition:

Tag	Name	Usage	Page	Required
CMD	Command definition	Multiple	268	Yes

Example

The following source file markup contains a command table that defines the commands UPDATE, ADD, DELETE and SEARCH.

```
<!DOCTYPE DM SYSTEM>

<CMDTBL APPLID=conv>
  <CMD NAME=update>Upd<T>ate
    <CMDACT ACTION='alias add'>
  <CMD NAME=add>Add
    <CMDACT ACTION=setverb>
  <CMD NAME=delete>Del<T>ete
    <CMDACT ACTION=passthru>
  <CMD NAME=search>Search
    <CMDACT ACTION=passthru>
</CMDTBL>
```

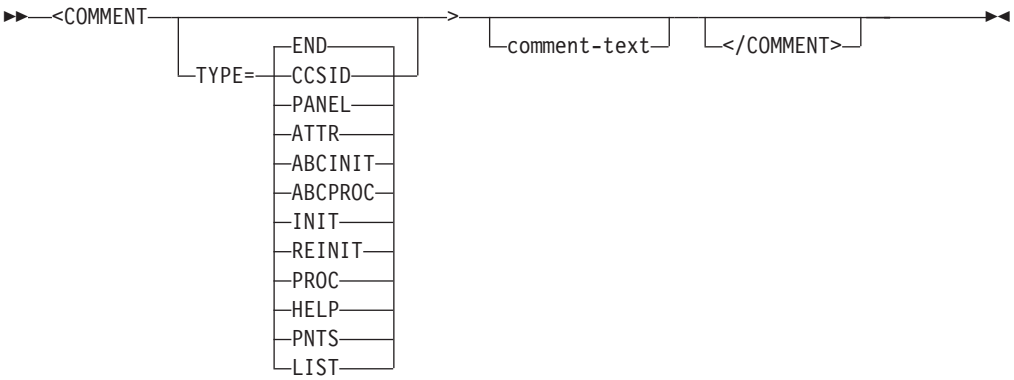
The following table shows the resultant ISPF application command table.

Table 3. ISPF Application Command Table

ZCTVERB	ZCTTRUNC	ZCTACT
UPDATE	3	ALIAS ADD
ADD	0	SETVERB
DELETE	3	PASSTHRU
SEARCH	0	PASSTHRU

COMMENT (Comment)

The COMMENT tag adds comment text to the generated panel or message member.



TYPE=END | CCSID | PANEL | ATTR | ABCINIT | ABCPROC | INIT | REINIT | PROC | HELP | PNTS | LIST

This attribute specifies the section of the panel that is to contain the comment text. The default is END. TYPE=END is assumed if the COMMENT tag is used within the MSGMBR tag.

COMMENT tags that specify the TYPE as ABCINIT or ABCPROC must follow an ABC or PDC tag.

When a COMMENT tag is coded within a HELP panel, the TYPE value is limited to CCSID, PANEL, ATTR, INIT, PROC, or END.

comment-text

The *comment-text* is flowed to a width of 66 bytes. The conversion utility adds "/* " before and " */" after the resulting text.

When no *comment-text* is present, a blank comment line is added to the specified (or defaulted) panel section.

Description

The COMMENT tag adds comments to the generated ISPF format panel. If the PREP conversion option has been specified, the comments will not be part of the final panel because they will not be processed by the ISPPREP utility.

Lines of text from a COMMENT tag are added to the specified panel section when encountered in the DTL source file.

Note: If the panel section specified will not be generated by other conversion processing, comments will be formatted as follows:

TYPE	Comments will be placed:
CCSID	Following the)PANEL statement.
LIST	Before the)END statement.

Comments added to the)END panel section are placed following any entries from the COPYR tag and comments containing the ISPDTLC version number and panel creation date. Lines placed in the)END section of a HELP panel are added to each continuation HELP panel.

Conditions:

- You must code the COMMENT tag within an ABC, AREA, CHOICE, DA, DTACOL DTAFLD, HELP, LSTCOL, LSTFLD, LSTGRP, PANEL, PDC, REGION or SELFLD tag definition.

Nested Tags

None.

Example

The following source file markup contains a comment of several lines that will be placed after the)END panel statement. Figure 105 on page 284 shows portion of the ISPF format panel containing the formatted result.

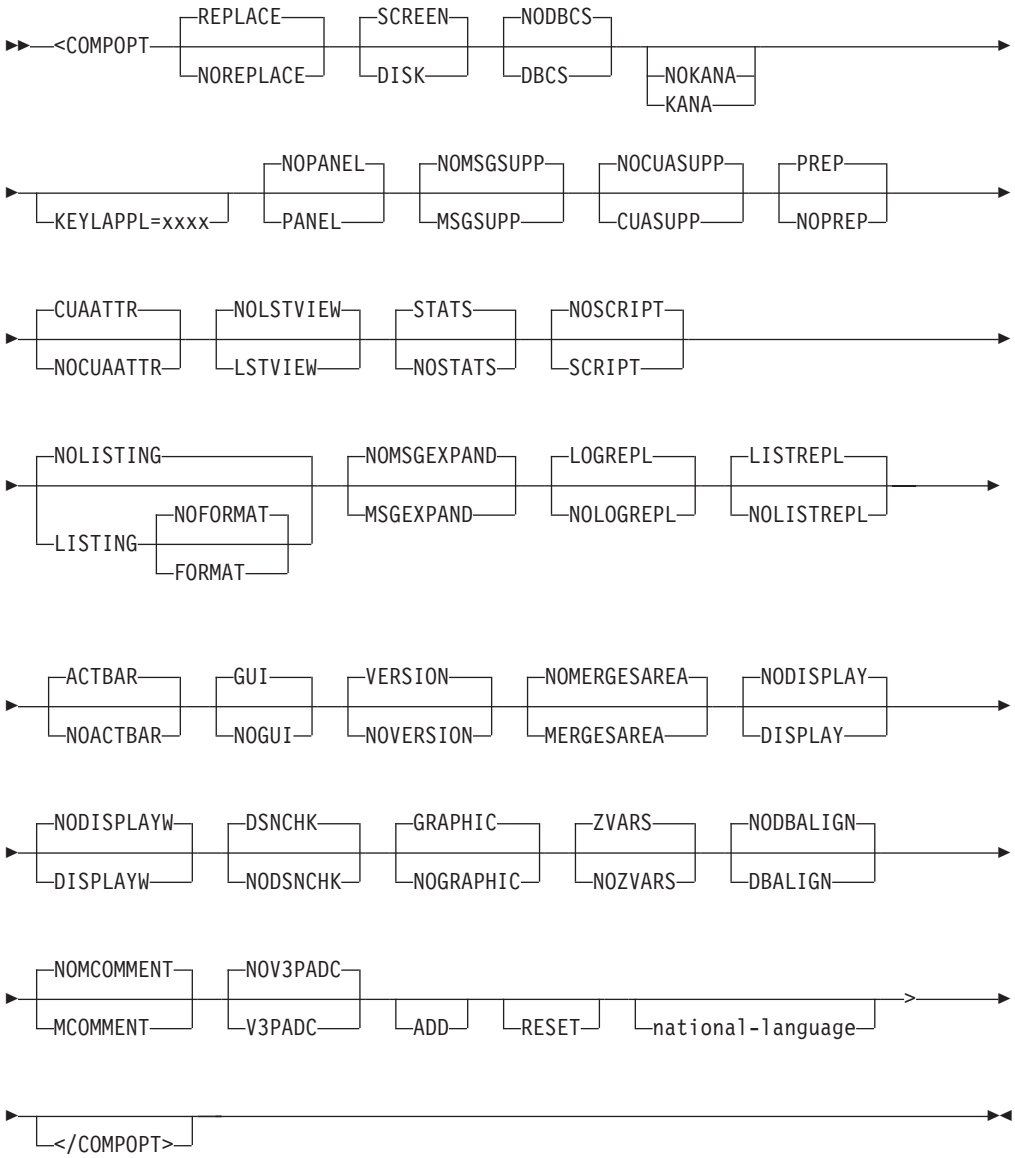
```
<!doctype dm system>
<!-- COMMENT tag example - PANEL tag -->
<!--   )END section - after CMDAREA tag  -->

<varclass name=vc1 type='char 10'>
<varclass name=vc2 type='char 6'>
<varlist>
<vardcl name=lst1 varclass=vc1>
<vardcl name=lst2 varclass=vc2>
</varlist>

<panel name=comment1 depth=19 width=50>
This is panel Comment1

<LSTFLD >
<LSTGRP headline=yes>
<LSTCOL colwidth=10 datavar=lst1 usage=in varclass=vc1 line=1
        required=yes autotab=yes align=end help=h1 msg=abcd101>COL1
<LSTCOL colwidth=6  datavar=lst2 usage=in varclass=vc2 line=2
        required=yes autotab=yes align=end help=h1 msg=abcd101>COL2
</LSTGRP>
</LSTFLD>
<cmdarea>
<comment type=end>
  comment line 1
    comment line 2
      comment line 3
  comment line 4
    comment line 5
      comment line 6
  comment line 7
    comment line 8
      comment line 9
</panel>
```

```
⋮

)END
/* comment line 1 comment line 2 comment line 3 comment line 4        */
/* comment line 5 comment line 6 comment line 7 comment line 8        */
/* comment line 9                                                     */
```

Figure 105. Comment text added to a panel

COMPOPT (Compiler Options)

The COMPOPT tag sets compiler options for the current source file.



With the exception of ADD and RESET, all of the option keywords used for the COMPOPT tag are the same as those used for "Conversion Utility Syntax" on page 183. Please refer to that chapter for information about these keywords.

The COMPOPT tag keyword RESET restores the conversion utility options to their original invocation settings.

Description

The COMPOPT tag can be placed within the Doctype definition to encompass the entire DTL source file or it can be placed before the first PANEL, HELP, MSGMBR, KEYL, or CMDTBL tag that requires a compiler option change.

Unless the ADD option is specified when the COMPOPT tag is processed, all conversion utility options except PANEL, DISK, SCREEN, DISPLAY, DISPLAYW, DBCS, and KANA are first reset to the defined default values. The options specified on the COMPOPT tag are then applied.

When the ADD option is included, the original options remain in effect and the options from the COMPOPT tag are added to the current list. ADD overrides any existing option.

The options set by this tag remain in effect for the current source file until another COMPOPT tag is processed. If you are converting a list of members, either from member list selections or from a DTLLST list of members, the conversion utility options are reset to their original invocation settings when the current source file is completed.

The PROFILE and PROFDDN options defined as part of the conversion utility invocation syntax are not supported by the COMPOPT tag.

Conditions:

None.

Nested Tags

None.

Example

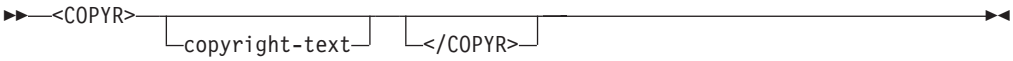
The following source file markup contains a compiler options line that specifies the compiler options to be used converting this source file.

```
<!doctype dm system>
<varclass name=vc1 type='char 10'>
<varclass name=vc2 type='char 6'>
<varlist>
<vardcl name=lst1 varclass=vc1>
<vardcl name=lst2 varclass=vc2>
</varlist>

<compopt noprep noreplace>

<panel name=compopt depth=19 width=50>
This is panel Compopt

<LSTFLD >
  <LSTGRP headline=yes>
    <LSTCOL colwidth=10 datavar=lst1 usage=in varclass=vc1 line=1
            required=yes autotab=yes align=end help=h1 msg=abcd101>COL1
    <LSTCOL colwidth=6  datavar=lst2 usage=in varclass=vc2 line=2
```

```
            required=yes autotab=yes align=end help=h1 msg=abcd101>COL2
  </LSTGRP>
</LSTFLD>
<cmdarea>
 </panel>
```

COPYR (Copyright)

The COPYR tag adds copyright text to the generated panel or message member.



copyright-text
The *copyright-text* is limited to 66 bytes. It is automatically formatted as a panel comment with a "/* " in front and a " */" following the supplied text.

Description

The COPYR tag adds copyright information to the panel.

The COPYR tag must be placed before the first PANEL, HELP, or MSGMBR definition within the DTL source file that is to contain the copyright information.

You can use multiple COPYR tags. Each tag creates one comment line, which is placed after the)END panel statement, or the last message in the message member, in the order found in the DTL source.

The *copyright-text* is added to each subsequent panel or message member generated from the same DTL source file member. If the PREP conversion option has been specified, the copyright will not be part of the final panel because comments are not processed by the ISPPREP utility.

Conditions

None.

Nested Tags

None.

Example

The following source file markup contains two copyright lines that will be placed after the)END panel statement. Figure 106 on page 287 shows a portion of the ISPF format panel containing the formatted result.

```
<!doctype dm system>
<!-- COPYR tag example - PANEL tag -->

<varclass name=vc1 type='char 10'>
<varclass name=vc2 type='char 6'>
<varlist>
<vardcl name=lst1 varclass=vc1>
<vardcl name=lst2 varclass=vc2>
</varlist>

<copyr>Copyright statement 1
<copyr>Copyright statement 2
```

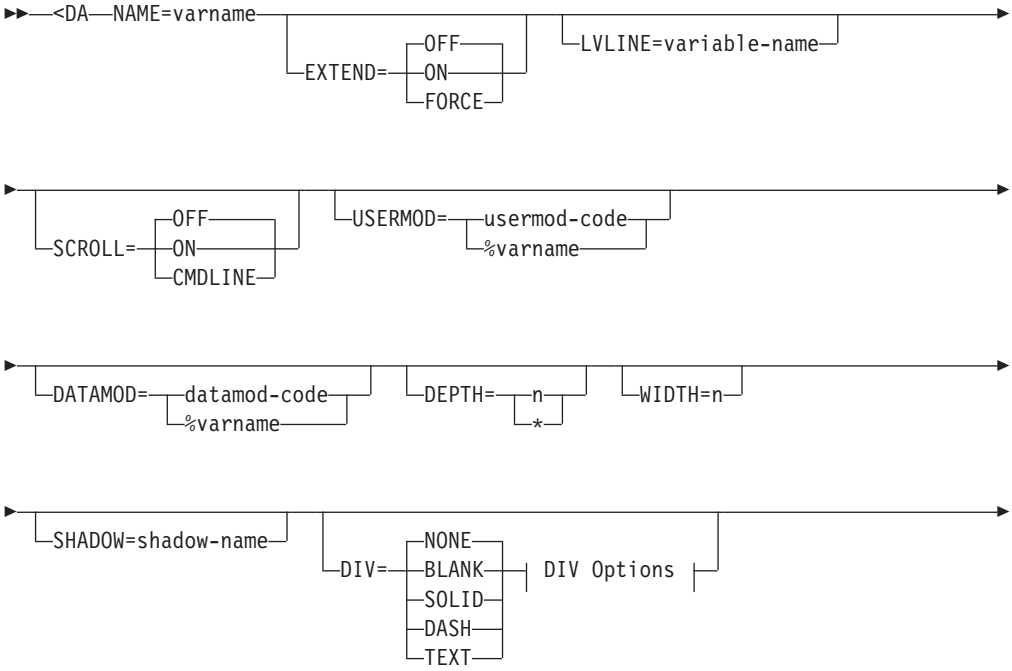
```
<panel name=copyrt1 depth=19 width=50>
This is panel Copyrt1

<LSTFLD >
<LSTGRP headline=yes>
<LSTCOL colwidth=10 datavar=lst1 usage=in varclass=vc1 line=1
        required=yes autotab=yes align=end help=h1 msg=abcd101>COL1
<LSTCOL colwidth=6  datavar=lst2 usage=in varclass=vc2 line=2
        required=yes autotab=yes align=end help=h1 msg=abcd101>COL2
</LSTGRP>
</LSTFLD>
<cmdarea>
</panel>
```

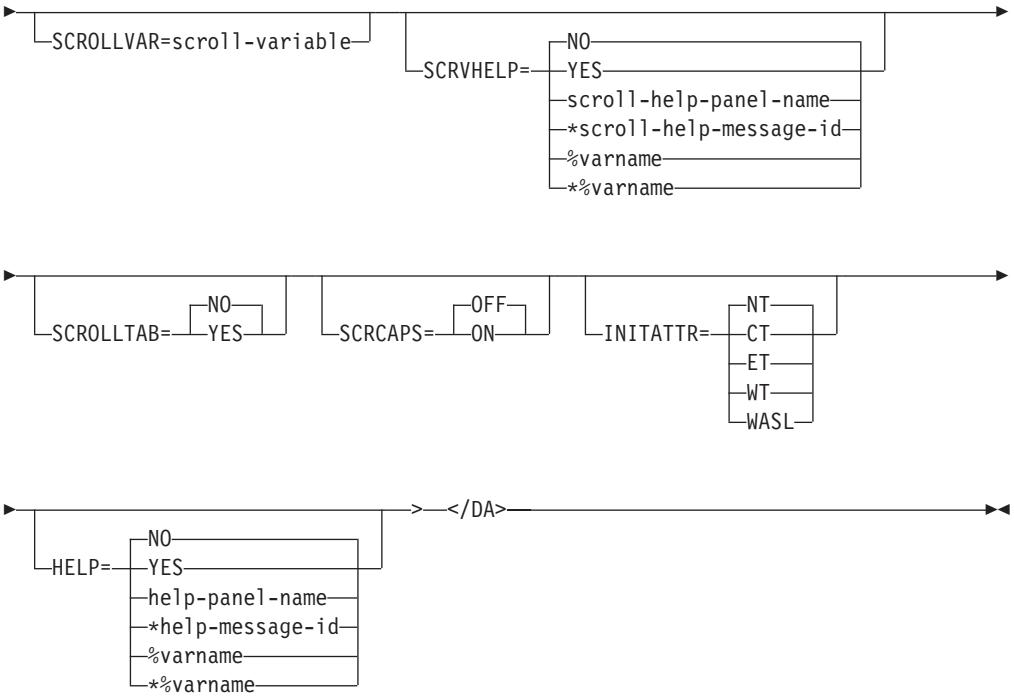
```
⋮

)END
/* Copyright statement 1 */
/* Copyright statement 2 */
```

Figure 106. Copyright statement added to a panel

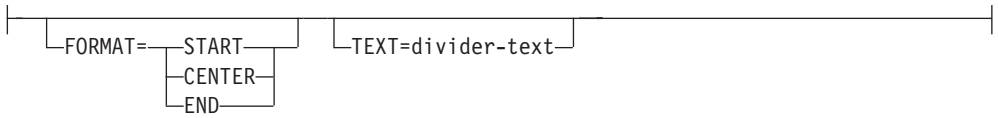
DA (Dynamic Area)

The DA tag defines a dynamic area in the panel)BODY section.





DIV Options:



NAME=varname
This attribute defines the name of a dynamic area. This name is the dialog variable specified by the application that contains the data for the dynamic area. The *varname* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

EXTEND=OFF | ON | FORCE
This attribute defines the run-time display size of the dynamic area. If EXTEND=ON is specified, the dynamic area definition is expanded to the size of the logical screen. If you intend to display the panels in a pop-up window, use EXTEND=OFF (which is the default.)

If EXTEND=FORCE is specified within a horizontal area or region, the EXTEND(ON) keyword is added to the dynamic area attribute statement in the)ATTR panel section. The conversion utility issues a message to advise of a potential display error if other panel fields are formatted on or after the last defined line of the dynamic area.

LVLINE=variable-name
This attribute allows you to specify the name of a variable that contains the result of the ISPF function LVLINE. The *variable-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

SCROLL=OFF | ON | CMDLINE
If you specify SCROLL=ON or SCROLL=CMDLINE, ISPDTLC adds the scroll amount field provided by the SCROLLVAR attribute to the command line.

If you specify SCROLL=ON, ISPDTLC also automatically enables scrolling commands by adding SCROLL(ON) to the dynamic area attribute definition.

Note: When SCROLL(ON) is *not* part of the dynamic area attribute definition, data in the scroll amount field is available to the application exactly as entered.

The first dynamic area on a panel that specifies SCROLL=ON or SCROLL=CMDLINE (with a valid SCROLLVAR attribute) controls the creation of the scroll amount field. The specification of the SCROLL attribute on subsequent DA tags is ignored.

USERMOD=usermod-code | %varname
This attribute specifies a single-character or a 2-position hexadecimal value to be substituted for attribute characters in a dynamic area variable following user interaction. You can define this attribute as a variable name preceded by a "%".

DATAMOD=datamod-code | %varname
This attribute specifies a single-character or a 2-position hexadecimal value to be substituted for attribute characters in a dynamic area following user interaction. You can define this attribute as a variable name preceded by a "%".

DEPTH=n | *
This attribute specifies the number of lines reserved for the dynamic area definition.

If the DA tag is to be formatted in the panel)BODY section, that is, the tag is not within a scrollable area:

- The maximum DEPTH value is the DEPTH value specified on the PANEL tag, reduced by the number of divider lines (if the DIV attribute is specified) and any other lines previously used by text or interactive fields.
- If the DEPTH value is specified as an asterisk (*), the conversion utility will reserve the remaining available panel depth for the dynamic area.

If the DA tag is defined within a scrollable area (see "AREA (Area)" on page 223), * cannot be specified as the depth value. The maximum DEPTH value is limited by the ISPF run-time environment.

WIDTH=n
This attribute specifies the number of columns reserved in the panel)BODY section for the dynamic area definition. If the dynamic area width is less than the PANEL width, the conversion utility adds an attribute byte immediately following the right dynamic area boundary. The minimum width for a dynamic area is the length of *varname* plus two (2) positions. The maximum value is the remaining panel width.

SHADOW=shadow-name
This attribute provides a name for a shadow variable name which is used to define character level attributes within the dynamic area string. The *shadow-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

DIV=NONE | BLANK | SOLID | DASH | TEXT
This attribute specifies the type of divider line to be placed before and after the

dynamic area. If this attribute is not specified or has the value NONE, no divider line is generated. The value BLANK produces a blank line. You must specify SOLID, DASH, or TEXT to produce a visible divider line. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line. A visible divider line formats with a non-displayable attribute byte on each end of the line.

FORMAT=START | CENTER | END
This attribute specifies the position of the *divider-text* within the divider line. You must specify both the FORMAT attribute and the TEXT attribute to create a divider line containing text.

TEXT=divider-text
This attribute specifies the text to be placed on the divider line. You must specify both the FORMAT attribute and the TEXT attribute to create a divider line containing text.

SCROLLVAR=scoll-variable
This attribute specifies the name of a variable that the application uses to obtain scrolling information. The *scroll-variable* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

SCRVHELP=NO | YES | scroll-help-panel-name | *scroll-help-message-id | %varname | *%varname
This attribute specifies the help action taken when the user requests help for the field specified with the SCROLLVAR attribute.

When SCRVHELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help on a choice and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *scroll-help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

SCROLLTAB=NO | YES
If you specify SCROLLTAB=YES, the cursor moves to the next input field when the user enters the last character in the scroll amount field. If there is no other input field on the panel, the cursor moves to the beginning of the command line. The ISPF SKIP keyword is not supported in GUI mode.

SCRCAPS=OFF | ON
When SCRCAPS=ON, the data in the scroll field is displayed in uppercase characters.

INITATTR=NT | CT | ET | WT | WASL
This attribute specifies the last attribute found before the start of the dynamic area. This allows the developer control of the initial color for the area. The conversion utility will replace the last attribute found before the dynamic area with the attribute specified.

HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname

This attribute specifies whether or not help is available for the dynamic area.

When HELP=YES, requesting help when the cursor is within the dynamic area causes control to return to the application. It is the application's responsibility to process the help request. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help in a dynamic area and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

Description

The DA tag defines a dynamic area in the panel)BODY or)AREA sections.

If you specify the CMDAREA tag within your DTL source file, it must appear before the DA tag when DEPTH=* is specified. The DA tag DEPTH may have to be adjusted to allow for additional lines which result from tags present within the panel definition following the end DA tag.

Refer to the *ISPF User's Guide* for a discussion of the Dynamic area in ISPF panels.

Conditions

- You must code the DA tag within a PANEL, AREA, or REGION tag. If found anywhere else, an error is logged and the output panel is not saved.
- If NAME is not valid or not specified, an error is logged and the output panel is not saved.
- You can use the EXTEND=ON attribute only once within a panel, and EXTEND=ON cannot be specified on a DA tag coded within a scrollable area. If EXTEND is already active, either from a DA tag, or from an AREA, GA, SELFLD or REGION tag, a warning message is logged and the EXTEND attribute is ignored.
- You can use the SCROLLVAR attribute only once within a panel.
- If you specify the SCROLLVAR attribute, you must also specify the attribute SCROLL=ON or SCROLL=CMDLINE.
- The resulting scroll entry on the command line must leave at least eight positions for the command entry field.
- If you specify the SCRVHELP attribute, you must also specify the SCROLLVAR attribute.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested Tags

You can code the following tags within a DA definition:

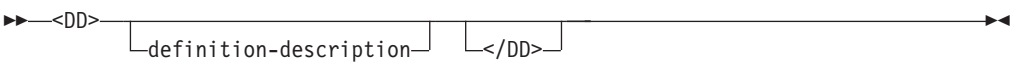
Tag	Name	Usage	Page	Required
ATTR	Attribute	Multiple	234	No
COMMENT	Comment	Multiple	282	No
SOURCE	Source	Multiple	472	No

Example

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampabc system>)>
&sampvar1;

<PANEL NAME=da KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
<TOPINST> Type in patron's name and card number (if applicable)
<AREA>
    <DTACOL PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25 SELWIDTH=25>
    <DTAFLD DATAVAR=curdate USAGE=out ENTWIDTH=8>Date
    <DTAFLD DATAVAR=cardno ENTWIDTH=7>Card No.
      <DTAFLDD>(A 7-digit number)
    <DTAFLD DATAVAR=name>Name
      <DTAFLDD>(Last, First, M.I.)
    <DTAFLD DATAVAR=address>Address
   </DTACOL>
  <DIVIDER>
  <DA NAME=darea DIV=solid DEPTH=6 SHADOW=shadwvar>
    <ATTR ATTRCHAR=#  TYPE=datain    PADC='_'  COLOR=BLUE>
    <ATTR ATTRCHAR=|  TYPE=dataout   COLOR=green>
    <ATTR ATTRCHAR=$  TYPE=char      COLOR=red>
  </DA>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

DD (Definition Description)

The DD tag defines the description of a term in a definition list.



definition-description
This is the text for the description of a definition list term.

Description

The DD tag defines the description of a term in a definition list.

Conditions

- You must code the DD tag within a DL definition. See "DL (Definition List)" on page 298 for a complete description of this tag.

- Each DD tag must follow an associated DT tag within the definition list. You can code only one DD tag for each DT tag.

Nested Tags

You can code the following tags within a DD definition:

Tag	Name	Usage	Page	Required
DL	Definition list	Multiple	298	No
FIG	Figure	Multiple	329	No
HP	Highlighted phrase	Multiple	354	No
LINES	Lines	Multiple	366	No
NOTE	Note	Multiple	400	No
NOTEL	Note List	Multiple	402	No
NT	Note	Multiple	404	No
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No
XMP	Example	Multiple	497	No

Example

The following help panel markup contains a definition list with three definition descriptions. Figure 107 on page 294 shows the formatted result.

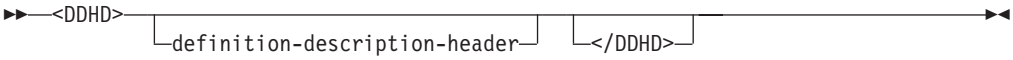
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=dd DEPTH=22 WIDTH=64>Help for Markup
<AREA>
<INFO>
  <P>Here are some definitions:
  <DL TSIZE=2 BREAK=all>
    <DT>markup
    <DD>Text that is added to document data in order to
    convey information about it.
    There are three types of markup the DTL uses:  tags, references,
    and markup declarations.
    <DT>markup declaration
    <DD>Markup that controls how other markup of a document
    is to be interpreted, for example document type and entity declarations.
    <DT>markup language
    <DD>A set of characters, conventions, and rules to control
    the interpretation of document data.
    The Dialog Tag Language is a markup language.
  </DL>
</INFO>
</AREA>
</HELP>
```

```
                        Help for Markup

Here are some definitions:

markup
  Text that is added to document data in order to convey
  information about it. There are three types of markup the
  DTL uses:  tags, references, and markup declarations.

markup declaration
  Markup that controls how other markup of a document is to be
  interpreted, for example document type and entity
  declarations.

markup language
  A set of characters, conventions, and rules to control the
  interpretation of document data. The Dialog Tag Language is
  a markup language.

 F1=Help         F3=Exit         F5=Exhelp       F6=Keyshelp
 F7=PrvTopic     F8=NxtTopic    F10=PrvPage     F11=NxtPage
F12=Cancel
```

Figure 107. Definition Descriptions

DDHD (Definition Description Header)

The DDHD tag defines the heading for the description column of a definition list.



definition-description-header
This is the text of the definition description header.

Description

The DDHD tag defines the heading for the description column of a definition list. You can code multiple DDHD tags within a definition list.

The conversion utility inserts a blank line between the header and the list items unless the COMPACT attribute is specified on the DL tag.

Conditions

- You must code the DDHD tag within a DL definition. See "DL (Definition List)" on page 298 for a complete description of this tag.
- Each DDHD tag must be paired with and follow a DTHD tag. See "DTHD (Definition Term Header)" on page 326 for a complete description of this tag.

Nested Tags

You can code the following tags within a DD definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No

Tag	Name	Usage	Page	Required
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

The following help panel markup contains a definition description header with the text “Meaning”. Figure 108 shows the formatted result.

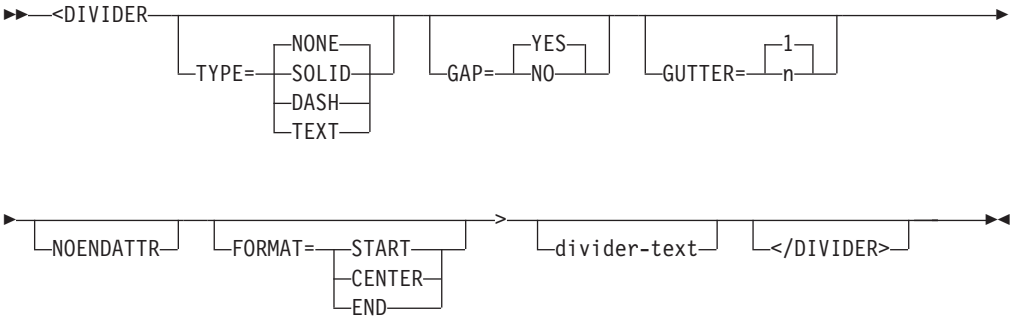
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=ddhd DEPTH=18>Prefix Help
<AREA>
<INFO>
  <P>The following list defines each of the valid prefixes.
  <DL TSIZE=12>
    <DTHD>Prefix
    <DDHD>Meaning
    <DT>AU
    <DD>Automotive
    <DT>HB
    <DD>Health and beauty
    <DT>LG
    <DD>Lawn and garden
    <DT>SG
    <DD>Sporting goods
  </DL>
</INFO>
</AREA>
</HELP>
```

```
                  Prefix Help

The following list defines each of the valid
prefixes.

Prefix      Meaning

AU          Automotive

HB          Health and beauty

LG          Lawn and garden

SG          Sporting goods

 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 108. Definition Description Header

DIVIDER (Area Divider)

The DIVIDER tag creates a blank or visible divider within the text portion of an application panel.



TYPE=NONE | SOLID | DASH | TEXT
This attribute specifies the type of divider line. The line width is one character.

The default value is NONE, which produces a blank line. You must specify SOLID, DASH, or TEXT to produce a visible divider line. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line.

GAP=YES | NO
When GAP=NO, the divider line completely crosses from one side of the text area to the other. When GAP=YES, a one-character gap remains at each end of the divider line. However, GAP=YES is ignored and set to NO for dividers coded within horizontal regions.

GUTTER=1 | n
This attribute specifies the total width of the divider. If the GUTTER value is an even number, the conversion utility increases the number by 1 so that the divider is centered within the defined width.

The minimum GUTTER value is 1. If GUTTER=1 on a DIVIDER within a horizontal region, then the TYPE value must be NONE.

The default GUTTER value for a DIVIDER within a vertical region is 1. The default GUTTER value for dividers within horizontal regions is 3 to allow for an attribute byte on each side of the divider character.

NOENDATTR
This attribute is valid only when the DIVIDER tag is coded within a horizontal region. It specifies that no ending attribute character will be placed after the divider character.

Note: The minimum divider space that can be specified for a horizontal region is 1.

When the GUTTER value is 1, the divider character is set to blank.

When the GUTTER value is 2, a solid divider may be specified. The divider character is placed in the second position of the 2 character GUTTER space.

FORMAT=START | CENTER | END
This attribute specifies the position of the divider text within the width of the divider line.

divider-text
This is the text of the area divider line.

Description

The DIVIDER tag creates a blank or solid divider within the text portion of an application panel. A horizontally formatted visible divider is created when you specify the TYPE attribute value as SOLID or DASH. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line. A vertically formatted SOLID or DASH divider is the "|" character which is obtained from the ISPF literals table. The direction of the divider is determined by the tag definition it is coded within. Formatting for dividers are as follows:

- Dividers coded within an AREA, HELP, or PANEL tag definition format horizontally.
- Dividers coded within a vertical region format horizontally.
- Dividers coded within a horizontal region format vertically.

The divider line can be formatted with descriptive text. When this feature is used, the FORMAT attribute must be specified. If FORMAT is not specified, the tag text is ignored. You control the text padding with the TYPE attribute. If TYPE=TEXT, the *divider-text* is padded with blanks. When TYPE=SOLID or TYPE=DASH, the *divider-text* is padded with the specified character.

Conditions

- You must code the DIVIDER tag within an AREA, DTACOL, HELP, PANEL, or REGION definition. See "AREA (Area)" on page 223, "DTACOL (Data Column)" on page 306, "HELP (Help Panel)" on page 340, "PANEL (Panel)" on page 415, and "REGION (Region)" on page 445 for descriptions of these tags.

Nested Tags

You can code the following tags within a DIVIDER definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No

Example

The following application panel markup contains four DIVIDER definitions. The first divider is blank. The second divider is solid with a gutter size of 2 and a GAP=NO value. The third and fourth dividers are solid. Figure 109 on page 298 shows the formatted result.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar3 system>)>
&sampvar3;

<PANEL NAME=divider DEPTH=22 WIDTH=70>Print a Document
<AREA>
  <DTACOL PMTWIDTH=20 ENTWIDTH=8 SELWIDTH=40 DESWIDTH=35>
    <DTAFLD DATAVAR=file>File name
      <DTAFLDD>Name of the document to be printed
    <DIVIDER TYPE=none>
```

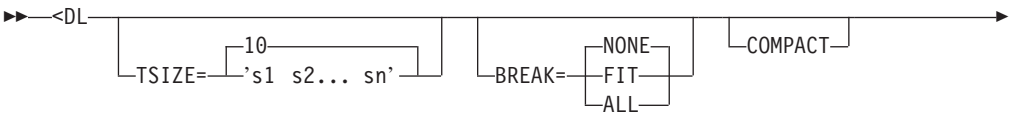
```
    <SELFLD NAME=type PMTLOC=before>Type style for printing
      <CHOICE>Prestige Elite (12 pitch)
      <CHOICE>Courier (10 pitch)
      <CHOICE>Essay Standard (proportional)
      <CHOICE>Essay Bold (proportional)
    </SELFLD>
  </DTACOL>
  <DIVIDER TYPE=solid GUTTER=2 GAP=no>
  <DTACOL PMTWIDTH=20 ENTWIDTH=2 DESWIDTH=35>
    <DTAFLD DATAVAR=marg>Left margin
      <DTAFLDD>Number of spaces in the left margin
    <DIVIDER TYPE=solid>
    <DTAFLD DATAVAR=copy>Copies
      <DTAFLDD>Number of copies
    <DIVIDER TYPE=solid>
    <DTAFLD DATAVAR=duplx ENTWIDTH=1>Duplex
      <DTAFLDD>1 = Yes (Print both sides of paper)
      <DTAFLDD>2 = No (Print one side only)
  </DTACOL>
</AREA>
</PANEL>
```

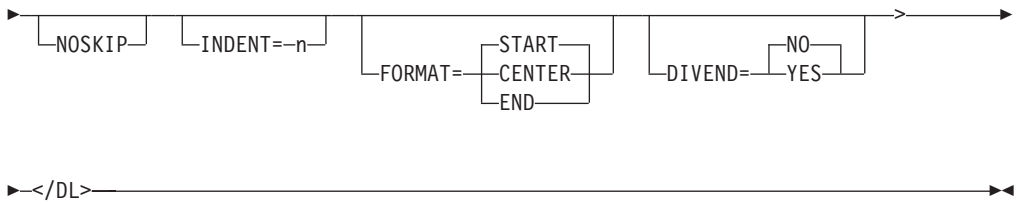
```
                          Print a Document

File name  . . . . . ________  Name of the document to be printed

Type style for
printing . . . . . __  1.  Prestige Elite (12 pitch)
                       2.  Courier (10 pitch)
                       3.  Essay Standard (proportional)
                       4.  Essay Bold (proportional)

------------------------------------------------------------------------

Left margin  . . . . __  Number of spaces in the left margin
 ----------------------------------------------------------------------
Copies . . . . . __  Number of copies
 ----------------------------------------------------------------------
Duplex . . . . . _  1 = Yes (Print both sides of paper)
                    2 = No (Print one side only)



 F1=Help    F3=Exit   F12=Cancel
```

Figure 109. Area Dividers

DL (Definition List)

The DL tag defines a list of terms and their corresponding definitions within an information region.





TSIZE=10 | 's1 s2... sn'
This attribute specifies the space to be allocated for the definition term. The default value is 10 characters. The minimum TSIZE value is 0 and the maximum is 40. When multiple TSIZE values are specified, a DT tag must be coded for each value. The sizes are applied to the DT tags in the order the tags are encountered in the DTL source file.

BREAK=NONE | FIT | ALL
This attribute controls the formatting of the definition terms and descriptions. If BREAK=NONE, the term is on the same line as the description, spilling into the description area if the length exceeds TSIZE. If BREAK=FIT, the description is on the line below the term if the term exceeds the TSIZE value. If BREAK=ALL, every definition is on the line below the term.

COMPACT
This attribute causes the list to format without a blank line between the items in the list. If you code DDHD and DTHD tags in a compact definition list, the list formats without a blank line between the headers and list items.

NOSKIP
This attribute causes the list to format without creating a blank line before the first line of the list.

INDENT=n
This attribute specifies that the definition list is to be indented from the current left margin.

FORMAT=START | CENTER | END
This attribute specifies the placement of the DT tag text within the space specified by TSIZE. The DL tag FORMAT setting applies to all of the DT tags within the definition list.

DIVEND=NO | YES
This attribute specifies a divider character will be formatted following the DDHD and DD tag text. When DIVEND=YES the formatting width of the DDHD and DD text is reduced to allow space for the divider character.

Description

The DL tag defines a list of terms and their corresponding definitions within an information region. You use the DT and DD tags to identify the terms that you are defining and their descriptions, respectively. You use the DTHD and the DDHD tags to define headings for the term and description columns in definition lists.

The conversion utility inserts a blank line before the definition list unless NOSKIP is specified.

If you do not specify a TSIZE value, the space allocated for the term size is 10 characters. If any term is longer than 10 characters and BREAK=NONE (the default) is specified, the term will extend into the description line. If the term is still too long to fit, it will wrap to the next line.

The definition description is an implied paragraph, and can contain any text items. For example, you can insert additional paragraphs in a definition description by using the paragraph (P) tag following the description paragraph. Other tags that you want to nest within the definition list (such as OL, SL, or UL) must follow the DD tag within the list.

Conditions

- The DL tag requires an end tag.
- You must code the DL tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.
- If you code DDHD and DTHD tags within the definition list, they must precede the first DT tag.

Nested Tags

You can code the following tags within a DL definition:

Tag	Name	Usage	Page	Required
DD	Definition description	Multiple	292	No
DDHD	Definition description header	Multiple	294	No
DLDIV	Definition List Divider	Multiple	302	No
DT	Definition term	Multiple	304	No
DTDIV	Definition Term Divider	Multiple	324	No
DTHD	Definition term header	Multiple	326	No
DTHDIV	Definition Term Header Divider	Multiple	328	No

Example

The following help panel markup contains a definition list that uses the default BREAK value of NONE, which formats the definition descriptions on the same line as the associated terms. Definition term and description headers are also included. Figure 110 on page 301 shows the formatted result of the markup. Figure 111 on page 301 shows how the same definition list would format with a BREAK value of FIT. Figure 112 on page 302 shows how the same definition list would format with a BREAK value of ALL.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=dl DEPTH=22 WIDTH=60>Employee Code Help
  <AREA>
  <INFO>
    <P>The following list defines the valid employee codes.
    <DL TSIZE=11>
      <DTHD>Code
      <DDHD>Meaning
      <DT>Full-time
      <DD>Indicates that the employee works a
      regular schedule of 40 hours or more weekly.
      <DT>Part-time
      <DD>Indicates that the employee works a regular
      schedule of 20 to 40 hours weekly.
      <DT>Supplemental
      <DD>Indicates that the employee works less than
      20 hours weekly.
      No regular schedule is in place.
```

```
    </DL>
  </INFO>
  </AREA>
</HELP>
```

```
                    Employee Code Help

The following list defines the valid employee codes.

Code       Meaning

Full-time  Indicates that the employee works a regular
           schedule of 40 hours or more weekly.

Part-time  Indicates that the employee works a regular
           schedule of 20 to 40 hours weekly.

Supplemental Indicates that the employee works less than
           20 hours weekly. No regular schedule is in
           place.




 F1=Help        F3=Exit        F5=Exhelp      F6=Keyshelp
 F7=PrvTopic    F8=NxtTopic   F10=PrvPage    F11=NxtPage
F12=Cancel
```

Figure 110. Definition List (BREAK=NONE)

```
                    Employee Code Help

The following list defines the valid employee codes.

Code       Meaning

Full-time  Indicates that the employee works a regular
           schedule of 40 hours or more weekly.

Part-time  Indicates that the employee works a regular
           schedule of 20 to 40 hours weekly.

Supplemental
           Indicates that the employee works less than 20
           hours weekly. No regular schedule is in place.




 F1=Help        F3=Exit        F5=Exhelp      F6=Keyshelp
 F7=PrvTopic    F8=NxtTopic   F10=PrvPage    F11=NxtPage
F12=Cancel
```

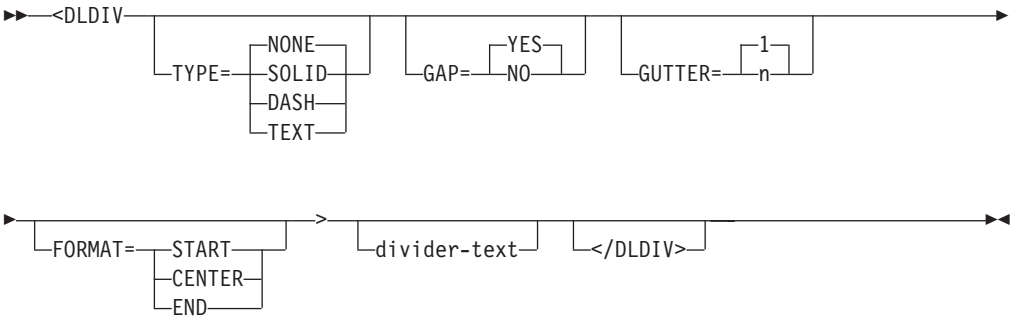
Figure 111. Definition List (BREAK=FIT)

```
                    Employee Code Help

The following list defines the valid employee codes.

Code       Meaning

Full-time
           Indicates that the employee works a regular
           schedule of 40 hours or more weekly.

Part-time
           Indicates that the employee works a regular
           schedule of 20 to 40 hours weekly.

Supplemental
           Indicates that the employee works less than 20
           hours weekly. No regular schedule is in place.


 F1=Help        F3=Exit        F5=Exhelp      F6=Keyshelp
 F7=PrvTopic    F8=NxtTopic   F10=PrvPage    F11=NxtPage
F12=Cancel
```

Figure 112. Definition List (BREAK=ALL)

DLDIV (Definition List Divider)

The DLDIV tag creates a blank or visible divider within the text portion of an application panel.



TYPE=NONE | SOLID | DASH | TEXT
This attribute specifies the type of divider line. The line width is one character.

The default value is NONE, which produces a blank line. You must specify SOLID, DASH, or TEXT to produce a visible divider line. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line.

GAP=YES | NO
When GAP=NO, the divider line completely crosses from one side of the text area to the other. When GAP=YES, a one-character gap remains at each end of the divider line.

GUTTER=1 | n
This attribute specifies the total width of the divider. If the GUTTER value is an even number, the conversion utility increases the number by 1 so that the divider is centered within the defined width.

The minimum GUTTER value, and the default, is 1.

FORMAT=START | CENTER | END
This attribute specifies the position of the divider text within the width of the divider line.

divider-text
This is the text of the area divider line.

Description

The DLDIV tag creates a blank or solid divider within the text portion of an application panel. A horizontally formatted visible divider is created when you specify the TYPE attribute value as SOLID or DASH. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line.

The divider line can be formatted with descriptive text. When this feature is used, the FORMAT attribute must be specified. If FORMAT is not specified, the tag text is ignored. You control the text padding with the TYPE attribute. If TYPE=TEXT, the *divider-text* is padded with blanks. When TYPE=SOLID or TYPE=DASH, the *divider-text* is padded with the specified character.

Conditions

- You must code the DLDIV tag within a DL tag.

Nested Tags

You can code the following tags within a DLDIV definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No

Example

The following example illustrates the use of the DLDIV tag in combination with the multiple DT tag function and the DIVEND attribute of the DL tag. Figure 113 on page 304 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=dldiv DEPTH=22 WIDTH=60>Employee Code Help
  <AREA>
  <INFO>
    <P>The following list defines the valid employee codes.
    <DL TSIZE=14 BREAK=none>
      <DLDIV TYPE=SOLID>
      <DTHD>Code
      <DDHD>Meaning
      <DLDIV TYPE=SOLID>
      <DT NOSKIP>Full-time
      <DD>Indicates that the employee works a
      regular schedule of 40 hours or more weekly.
      <DT>Part-time
      <DD>Indicates that the employee works a regular
```

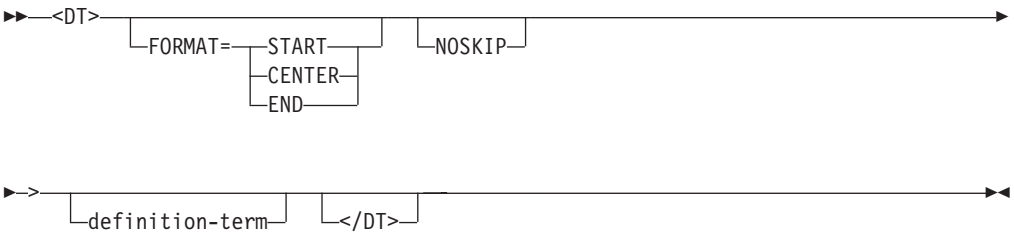
```
      schedule of 20 to 40 hours weekly.
      <DT>Supplemental
      <DD>Indicates that the employee works less than
      20 hours weekly.
      No regular schedule is in place.
    </DL>
  </INFO>
  </AREA>
</HELP>
```

```
                    Employee Code Help

The following list defines the valid employee codes.
------------------------------------------------------------
Code          Meaning
------------------------------------------------------------
Full-time     Indicates that the employee works a regular
              schedule of 40 hours or more weekly.

Part-time     Indicates that the employee works a regular
              schedule of 20 to 40 hours weekly.

Supplemental  Indicates that the employee works less than
              20 hours weekly. No regular schedule is in
              place.
```

Figure 113. Definition List Dividers

DT (Definition Term)

The DT tag defines a term in a definition list.



FORMAT=START | CENTER | END
This attribute specifies the placement of the DT tag text within the space specified by TSIZE. The DT tag FORMAT setting overrides the FORMAT specified in the enclosing DL tag.

NOSKIP
This attribute causes the definition term to be formatted without a blank line before the term. It is used to control the formatting of the definition term when COMPACT has not been specified on the enclosing DL tag. When the DL tag

TSIZE attribute specifies that multiple DT tags are to be formatted for each DD tag, NOSKIP should be coded on the first DT tag. It is ignored for the second and subsequent DT tags.

definition-term
This is the text of the definition term.

Description

The DT tag defines a term in a definition list.

Conditions

- You must code the DT tag within a DL definition. See "DL (Definition List)" on page 298 for a complete description of this tag.
- Each DT tag must be paired with and precede a DD tag.

Nested Tags

You can code the following tags within a DD definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

The following help panel markup contains a definition list with three definition terms. Each definition term is paired with an associated definition description. Figure 114 on page 306 shows the formatted result.

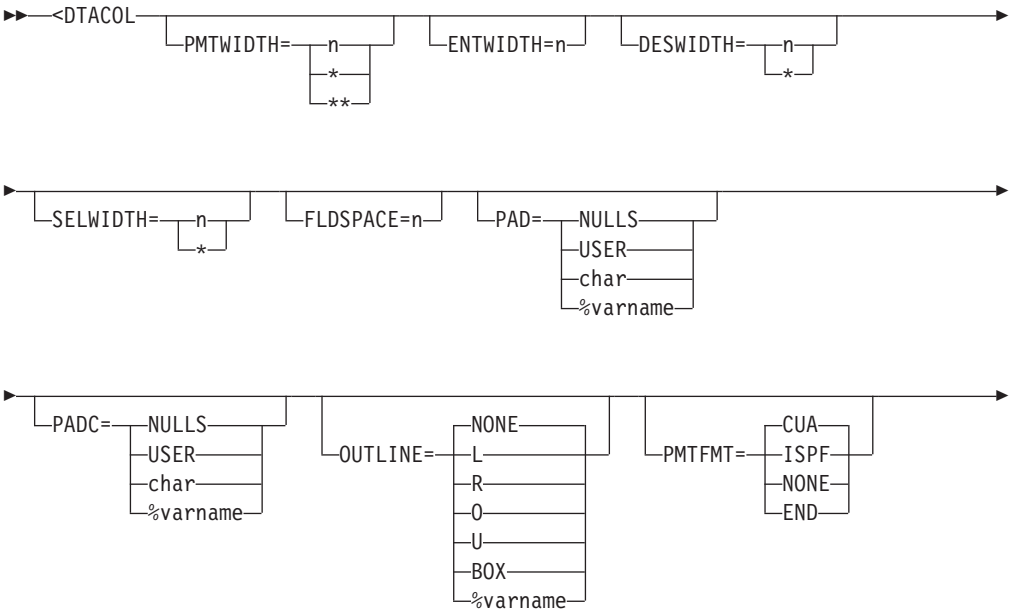
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=dt DEPTH=22 WIDTH=64>Help for Markup
<AREA>
<INFO>
  <P>Here are some definitions:
  <DL TSIZE=2 BREAK=all>
    <DT>markup
    <DD>Text that is added to document data in order to
    convey information about it.
    There are three types of markup the DTL uses:  tags, references,
    and markup declarations.
    <DT>markup declaration
    <DD>Markup that controls how other markup of a document
    is to be interpreted, for example document type and entity declarations.
    <DT>markup language
    <DD>A set of characters, conventions, and rules to control
    the interpretation of document data.
    The Dialog Tag Language is a markup language.
  </DL>
</INFO>
</AREA>
</HELP>
```

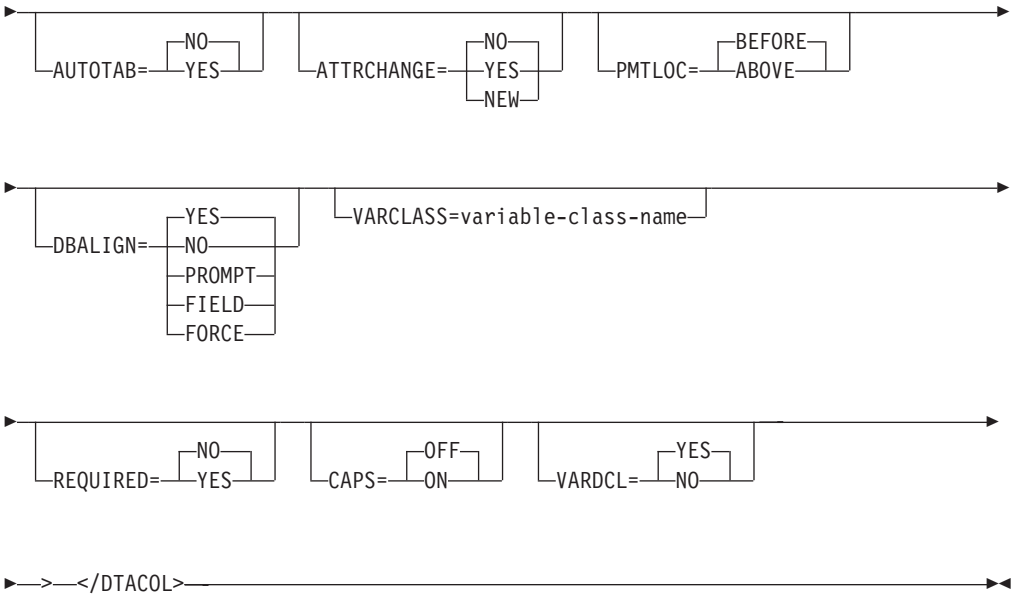
```
                        Help for Markup

Here are some definitions:

markup
  Text that is added to document data in order to convey
  information about it.  There are three types of markup the
  DTL uses: tags, references, and markup declarations.

markup declaration
  Markup that controls how other markup of a document is to be
  interpreted, for example document type and entity
  declarations.

markup language
  A set of characters, conventions, and rules to control the
  interpretation of document data. The Dialog Tag Language is
  a markup language.

 F1=Help        F3=Exit        F5=Exhelp      F6=Keyshelp
 F7=PrvTopic    F8=NxtTopic   F10=PrvPage    F11=NxtPage
F12=Cancel
```

Figure 114. Definition Terms

DTACOL (Data Column)

The DTACOL tag defines default values for data fields (DTAFLD) and selection fields (SELFLD) that are coded within a DTACOL definition.





PMTWIDTH=n | * | **
This attribute specifies the number of bytes reserved for prompts for data fields and selection fields coded within the data column. The minimum width is 0 and the maximum is the remaining available panel width. When you specify PMTWIDTH=*, the conversion utility uses the length of the prompt text as the prompt width. When you specify PMTWIDTH=**, the conversion utility uses the maximum available space as the prompt width. If PMTFMT=CUA is specified (or defaulted) and the prompt text has fewer characters than the field allows, leader dots fill the remaining spaces. For output-only data fields, a colon is also added as the last character in the prompt width space. If any prompt contains more characters than the width you specify, the prompt is word-wrapped to fit on multiple lines.

Note: Any field within the data column defining a prompt width overrides the DTACOL PMTWIDTH value.

ENTWIDTH=n
This attribute specifies the number of bytes reserved for data fields coded within the data column. The minimum width is 1 and the maximum is the remaining available panel (or region) width.

Note: Any data field within the data column defining an entry width overrides the DTACOL ENTWIDTH value.

DESWIDTH=n | *
This attribute specifies the number of bytes reserved for the description text of the enclosed DTAFLDD tags. The minimum width is 0. When you specify DESWIDTH=*, the conversion utility uses the length of the description text as the description width. If the text is longer than the width you specify, the text is word-wrapped to fit on multiple lines.

Note: Any data field within the data column defining a description width overrides the DTACOL DESWIDTH value.

SELWIDTH=n | *
This attribute specifies the number of bytes reserved for choices in selection

fields coded within the data column. The minimum width value is 1 and the maximum is the remaining available panel width. If the width required by the *choice-text* and its entry field exceeds the specified SELWIDTH value, the text is wrapped to multiple conversion utility will use the remaining available panel (or region) width.

Note: Any selection field within the data column defining a selection width overrides the DTACOL SELWIDTH value.

FLDSPACE=n
This attribute specifies the number of bytes reserved for the data field. The minimum width is 2 and the maximum is the remaining available panel (or region) width. The FLDSPACE value should include the actual entry width plus the number of entry field attributes. If the value specified by ENTWIDTH (plus attributes) is less than the specified FLDSPACE value, the entry field is padded with blanks to the FLDSPACE value. This will create blank space between the entry field and description text provided by the DTAFLDD tag and allows you to align description text from successive DTAFLD definitions.

Note: Any data field within the data column defining field space overrides the DTACOL FLDSPACE value.

PAD=NULLS | USER | char | %varname
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

Note: Any data field within the data column defining PAD overrides the DTACOL PAD value.

PADC=NULLS | USER | char | %varname
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

Note: Any data field within the data column defining PADC overrides the DTACOL PADC value.

OUTLINE=NONE | L | R | O | U | BOX | %varname
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

Note: Any data field within the data column defining OUTLINE overrides the DTACOL OUTLINE value.

PMTFMT=CUA | ISPF | NONE | END
This attribute controls the generation of prompt leader characters. The default is to create CUA leader dots. When ISPF is specified, and at least 4 bytes of prompt text space remain following the prompt text, the "===>" character string is placed in the rightmost 4 positions of the prompt text space. When NONE is specified, no leader characters are added to the prompt text. When END is specified, the prompt text is right justified within the prompt text space.

Note: Any data field within the data column defining PMTFMT overrides the DTACOL PMTFMT value.

AUTOTAB=NO | YES
When AUTOTAB=YES, the cursor moves to the next field capable of input when the user enters the last character in this field. If no other field capable of

user input exists on the panel, the cursor returns to the beginning of this field. The ISPF SKIP keyword is not supported when running in GUI mode.

Note: Any data field within the data column defining AUTOTAB overrides the DTACOL AUTOTAB value.

ATTRCHANGE=NO | YES | NEW
When ATTRCHANGE=YES or ATTRCHANGE=NEW, the conversion utility formats an additional entry in the panel)ATTR section (that can apply to multiple data fields) instead of creating a unique ".ATTR(field-name)" entry in the)INIT section for each field. With this option, multiple DTAFLD tags with the same characteristics require fewer panel logic statements. ATTRCHANGE=NEW creates a new entry. ATTRCHANGE=YES uses an existing attribute, if possible.

Note: Any data field within the data column defining ATTRCHANGE overrides the DTACOL ATTRCHANGE value.

PMTLOC=BEFORE | ABOVE
This attribute defines the prompt location for the enclosed DTAFLD and SELFLD tags.

Note: Any data field or selection field within the data column defining PMTLOC overrides the DTACOL PMTLOC value.

DBALIGN=YES | NO | PROMPT | FIELD | FORCE
This attribute defines the DBALIGN value for the enclosed DTAFLD tags.

Note: Any data field within the data column defining DBALIGN overrides the DTACOL DBALIGN value.

VARCLASS=variable-class-name
This attribute defines the name of the variable class for enclosed CHOFLD and DTAFLD tags.

Note: Any data field within the data column defining VARCLASS overrides the DTACOL VARCLASS value.

REQUIRED=NO | YES
This attribute defines whether the fields for enclosed CHOFLD and DTAFLD tags ***require*** input.

Note: Any data field within the data column defining REQUIRED overrides the DTACOL REQUIRED value.

CAPS=OFF | ON
This attribute defines whether the fields for enclosed CHOFLD and DTAFLD tags are displayed in uppercase characters.

Note: Any data field within the data column defining CAPS overrides the DATCOL CAPS value.

VARDCL=YES | NO
When VARDCL=NO the data field name is not checked to the declared variable information provided with the VARCLASS and VARDCL tags for enclosed CHOFLD and DTAFLD tags.

Note: Any data field within the data column defining VARDCL overrides the DATCOL VARDCL value.

Description

The DTACOL tag defines default attribute values for data fields (DTAFLD), choice data fields (CHOFLD), and selection fields (SELFLD) that are coded within a DTACOL definition. This allows you to define common values for fields coded within the data column within a single tag definition.

The xxxWIDTH attributes are convenient for aligning fields on an application panel. Fields are laid out within the data column along boundaries established by the values specified on the DTACOL tag. The following example shows those boundaries:

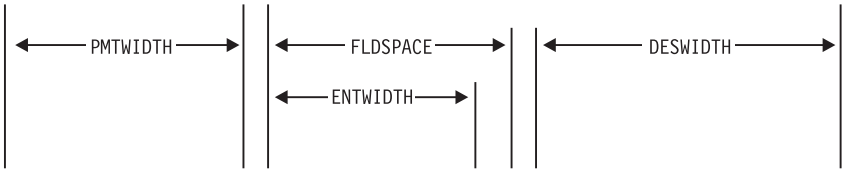


Figure 115.

The prompt width (PMTWIDTH) is valid for data fields and selection fields coded within the data column description. The entry width (ENTWIDTH), field space (FLDSPACE), and description width (DESWIDTH) are only used by enclosed DTAFLD tags. The selection width (SELWIDTH) is used only by enclosed SELFLD tags. All of the previous cases stated are true only when the enclosed DTAFLD or SELFLD tags do not specify values that override the DTACOL values.

Note: The SELFLD tag does not use the ENTWIDTH, DESWIDTH, FLDSPACE, PAD, PADC, OUTLINE, AUTOTAB, ATTRCHANGE, DBALIGN, VARCLASS, REQUIRED, or CAPS attributes of the DTACOL tag.

If the combined PMTWIDTH, ENTWIDTH, and DESWIDTH values exceed the remaining available panel (or region) width, the conversion utility issues a warning message and attempts to fit the data in the available width by wrapping the text.

For data fields, first priority is given to the entry field. Second and third priorities are given to the prompt and description fields, respectively. These fields use the available width remaining after the width of the entry field is determined.

Note: Word wrapping can result in word truncation if insufficient width is available for the text.

Conditions

- The DTACOL tag requires an end tag.
- You must code the DTACOL tag within an AREA, PANEL, or REGION definition. You can code a DTACOL definition anywhere within these tags, but the start and end tags must enclose any DTAFLD or SELFLD tags to which it applies. See “AREA (Area)” on page 223, “PANEL (Panel)” on page 415, and “REGION (Region)” on page 445 for descriptions of these tags.
- If both PAD and PADC have been specified, PAD is ignored and PADC is used.
- When a “%varname” notation is found on any of the attributes that allow a variable name, the “%varname” entry must follow the standard naming convention described in “Rules for “%variable” Names” on page 211.

Nested Tags

You can code the following tags within a DTACOL definition:

Tag	Name	Usage	Page	Required
COMMENT	Comment	Multiple	282	No
DIVIDER	Area divider	Multiple	296	No
DTAFLD	Data field	Multiple	312	No
GRPHDR	Group header	Multiple	337	No
SELFLD	Selection field	Multiple	455	No
SOURCE	Source	Multiple	472	No

Example

The following application panel markup contains a data column that provides default width values for the enclosed data fields and data field descriptions. The ENTWIDTH value specified on the first and second data fields override the ENTWIDTH value specified on the DTACOL tag. Figure 116 on page 312 shows the formatted result.

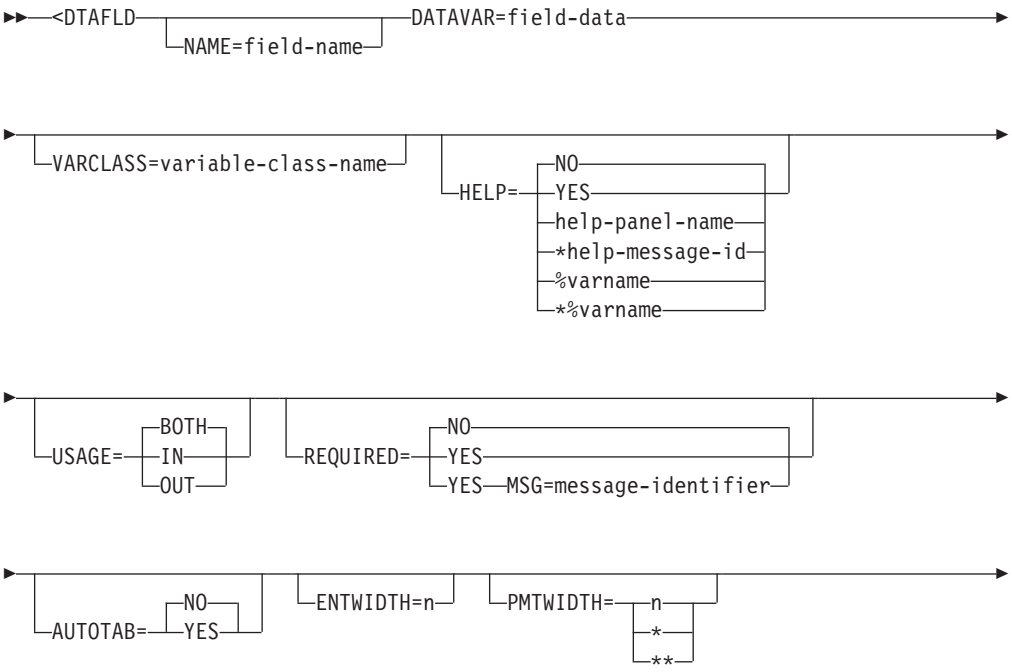
```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampabc system>)>
&sampvar1;

<PANEL NAME=dtacol2 KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
<TOPINST> Type in patron's name and card number (if applicable)
<TOPINST> Then select an action bar choice.
<AREA>
  <DTACOL PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25 SELWIDTH=25
    <DTAFLD DATAVAR=curdate USAGE=out ENTWIDTH=8>Date
    <DTAFLD DATAVAR=cardno ENTWIDTH=7>Card No.
      <DTAFLDD>(A 7-digit number)
    <DTAFLD DATAVAR=name>Name
      <DTAFLDD>(Last, First, M.I.)
    <DTAFLD DATAVAR=address>Address
  </DTACOL>
  <DIVIDER>
  <REGION DIR=horiz>
  <SELFLD NAME=cardsel PMTWIDTH=30 SELWIDTH=38>Choose
  one of the following
    <CHOICE CHECKVAR=card MATCH=new>New
    <CHOICE CHECKVAR=card MATCH=renew>Renewal
    <CHOICE CHECKVAR=card MATCH=replace>Replacement
  </SELFLD>
  <SELFLD TYPE=multi PMTWIDTH=30 SELWIDTH=25>Check valid branches
    <CHOICE NAME=north HELP=nthhlp CHECKVAR=nth>North Branch
    <CHOICE NAME=south HELP=sthhlp CHECKVAR=sth>South Branch
    <CHOICE NAME=east HELP=esthlp CHECKVAR=est>East Branch
    <CHOICE NAME=west HELP=wsthlp CHECKVAR=wst>West Branch
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

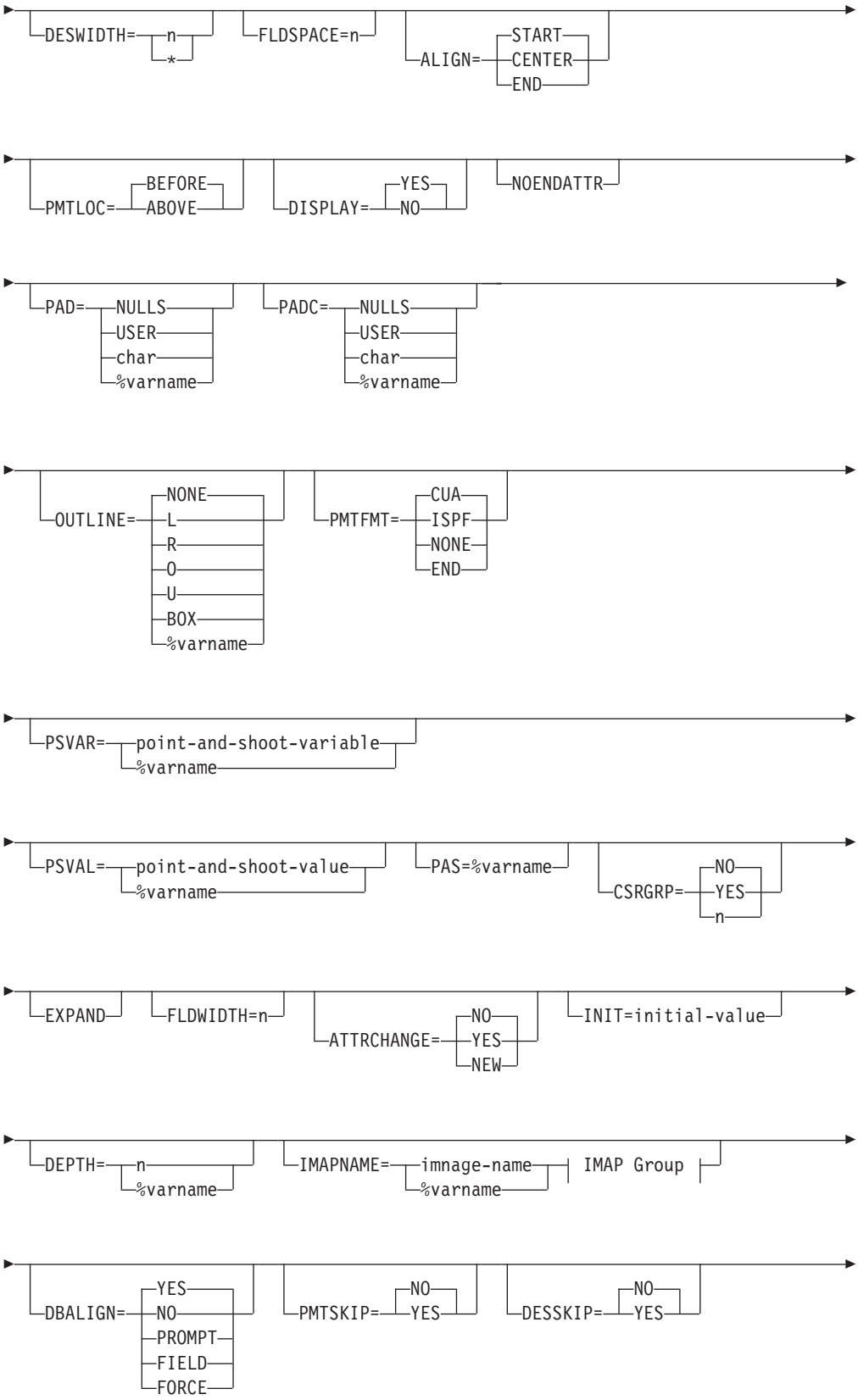
```
  File  Search  Help
------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number if applicable.

Then select an action bar choice.

Date . . . : 08/29/90
Card No. . . _______                    (A 7-digit number)
Name . . . . _____________________      (Last, First, M.I.)
Address  . . _____________________

Choose one of the following          Check valid branches
__  1.  New                         _  North Branch
    2.  Renewal                     _  South Branch
    3.  Replacement                 _  East Branch
                                    _  West Branch

Enter a command ===> ___________________________________________________________
 F1=Help        F2=Split       F3=Exit        F6=KEYSHELP    F9=Swap
F12=Cancel
```

Figure 116. Data Column

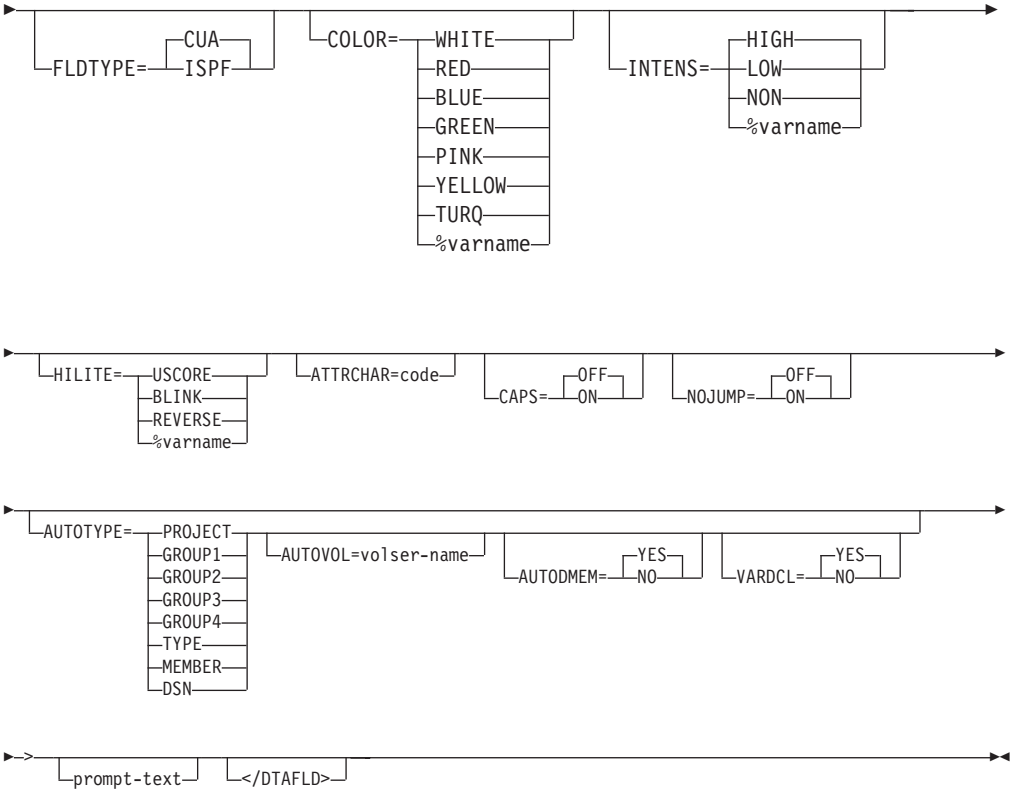
DTAFLD (Data Field)

The DTAFLD tag defines an input field, an output field, or an input/output field on an application panel.

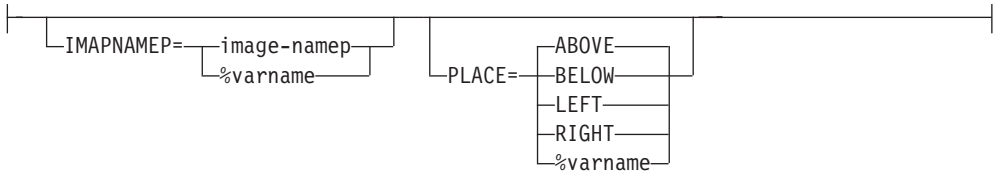



DESWIDTH=
n
*
FLDSPACE=n
ALIGN=
START
CENTER
END
PMTLOC=
BEFORE
ABOVE
DISPLAY=
YES
NO
NOENDATTR
PAD=
NULLS
USER
char
%varname
PADC=
NULLS
USER
char
%varname
OUTLINE=
NONE
L
R
O
U
BOX
%varname
PMTFMT=
CUA
ISPF
NONE
END
PSVAR=
point-and-shoot-variable
%varname
PSVAL=
point-and-shoot-value
%varname
PAS=%varname
CSRGRP=
NO
YES
n
EXPAND
FLDWIDTH=n
ATTRCHANGE=
NO
YES
NEW
INIT=initial-value
DEPTH=
n
%varname
IMAPNAME=
imnage-name
%varname
IMAP Group
DBALIGN=
YES
NO
PROMPT
FIELD
FORCE
PMTSKIP=
NO
YES
DESSKIP=
NO
YES

DTAFLD



IMAP Group:



NAME=field-name
This attribute specifies the name of the field. The *field-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

The *field-name* can be used by:
- The PANEL tag to position the cursor
- The ISPF DISPLAY or TBDISPL services to position the cursor
- The ISPF ADDPOP service to position a pop-up.

DATAVAR=field-data
This attribute specifies the variable name for the data in the field. The value coded must be a variable-name without the leading % notation. The conversion utility considers NAME and DATAVAR to be synonymous. However, the value you assign DATAVAR has precedence. For example, if you specify different values for the DATAVAR and NAME attributes, the conversion utility uses the DATAVAR value as the name of the field on the panel.

Compatibility Considerations

DATAVAR is a required attribute for the DTAFLD tag. For backward compatibility between releases, you can code either the NAME or the DATAVAR attributes, or both.

VARCLASS=variable-class-name
This attribute specifies the name of the variable class, defined using a VARCLASS tag, that overrides the default variable class referred to by the VARDCL that declared the data variable for this field.

HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname
This attribute specifies the help action taken when the user requests help for this data field. This is field-level help.

When HELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help for the data field and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

USAGE=BOTH | IN | OUT
This attribute indicates whether the field is for input only, output only, or both.

For USAGE=OUT, the conversion utility inserts a colon as the last character of the data field prompt to indicate to the user that it is an output-only field.

REQUIRED=NO | YES
This attribute indicates if the field requires input. This attribute is valid only when USAGE=IN or BOTH.

If REQUIRED=YES is coded, a VER(variable,NONBLANK) statement will be built by the conversion utility and placed in the)PROC section of the ISPF panel generated.

MSG=message-identifier
This attribute specifies the message that is displayed when the user does not complete a required entry (defined with the REQUIRED attribute). If you do not specify a *message-identifier*, ISPF displays a default message.

If you specify the MSG attribute and REQUIRED=YES, a VER(variable,NONBLANK,MSG=message-identifier) statement is built by the conversion utility and placed in the)PROC section of the ISPF panel generated. If you specify the MSG attribute and REQUIRED=NO (the default), the conversion utility issues a warning message.

See "MSG (Message)" on page 393 for information on creating messages.

Note: You can specify messages pertaining to other validations using XLATL and CHECKL tags within a VARCLASS definition. See the descriptions of these tags for additional information.

AUTOTAB=NO | YES
When AUTOTAB=YES, the cursor moves to the next field capable of input when the user enters the last character in this field. If no other field capable of user input exists on the panel, the cursor returns to the beginning of this field. The ISPF SKIP keyword is not supported when running in GUI mode.

AUTOTAB=YES is valid only when the value for USAGE is either BOTH or IN. If specified, this attribute overrides the AUTOTAB value of the DTACOL tag.

ENTWIDTH=n
This attribute specifies the number of bytes used for the data field. The minimum width is 1 and the maximum is the remaining available panel width less the required amount of space for field attributes. If ENTWIDTH is not provided on either the DTAFLD tag or the enclosing DTACOL tag, the conversion utility will use the width determined by the TYPE value of the associated VARCLASS.

If specified, this attribute overrides the ENTWIDTH value of the DTACOL tag.

**PMTWIDTH=n | * | **
This attribute specifies the number of bytes used for the data field *prompt-text*. The minimum width is 0 and the maximum is the remaining available panel (or region) width less the required amount of space for field attributes. When you specify PMTWIDTH=*, the conversion utility uses the length of the prompt text as the prompt width. When you specify PMTWIDTH=**, the conversion utility uses the maximum available space as the prompt width. If PMTFMT=CUA is specified (or defaulted) and the *prompt-text* has fewer characters than the field allows, leader dots fill the remaining spaces. If any prompt contains more characters than the width you specify, the prompt is word-wrapped to fit on multiple lines. If PMTWIDTH is not specified and *prompt-text* is present, the PMTWIDTH value will be defaulted to the length of the *prompt-text*.

If specified, this attribute overrides the PMTWIDTH value of the DTACOL tag.

DESWIDTH=n | *
This attribute specifies the number of bytes used for the description text of enclosed DTAFLDD tags. The minimum width is 0. When you specify DESWIDTH=*, the conversion utility uses the length of the description text as the description width. If the text is longer than the width you specify, the text is word-wrapped to fit on multiple lines.

If specified, this attribute overrides the DESWIDTH value of the DTACOL tag.

FLDSPACE=n
This attribute specifies the number of bytes reserved for the data fields coded within the data column. The minimum width is 2 and the maximum is the remaining available panel (or region) width. The FLDSPACE value should include the actual entry width plus the number of entry field attributes. If the value specified by ENTWIDTH is less than the specified FLDSPACE value, the entry field is padded with blanks to the FLDSPACE value. This will create blank space between the entry field and description text provided by the DTAFLDD tag and allows you to align description text from successive DTAFLD definitions.

If specified, this attribute overrides the FLDSPACE value of the DTACOL tag.

ALIGN=START | CENTER | END
This attribute specifies the alignment of data within the display field after all translations have been performed. Use this attribute to align the data with the start, the end, or the center of the display field.

This is accomplished in the conversion utility by using an attribute character for the field that specifies JUST(LEFT) if ALIGN=START or JUST(RIGHT) if ALIGN=END. ALIGN=CENTER will use an attribute character for the field that specifies JUST(ASIS).

Alignment occurs if the transformed value of the dialog variable is shorter than the display width of the field. When ALIGN=END, there is no underscore padding performed–blanks are used.

PMTLOC=BEFORE | ABOVE
This attribute specifies whether the *prompt-text* of the data field appears above or in front of the data field.

DISPLAY=YES | NO
This attribute specifies whether data will display on the screen when the user types it in. If you specify NO, the data will not display. This attribute is useful when creating fields for passwords or other information which you do not want to appear on the screen.

NOENDATTR
This attribute, which is valid only when WINDOW=NO is specified on the PANEL tag or DIR=HORIZ is specified on the REGION tag, specifies that no ending attribute will be placed after the data field. NOENDATTR is ignored for the last field on each panel line unless WINDOW=NO has been specified on the PANEL tag. NOENDATTR is valid only when the DTAFLD tag is followed by a DTAFLD, DTAFLDD, DIVIDER, or SELFLD tag.

PAD=NULLS | USER | char | %varname
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

If specified, this attribute overrides the PAD value of the DTACOL tag.

PADC= NULLS | USER | char | %varname
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

If specified, this attribute overrides the PADC value of the DTACOL tag.

OUTLINE=NONE | L | R | O | U | BOX | %varname
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

If specified, this attribute overrides the OUTLINE value of the DTACOL tag.

PMTFMT=CUA | ISPF | NONE | END
This attribute controls the generation of prompt leader characters. The default is to create CUA leader dots. When ISPF is specified, and at least 4 bytes of prompt text space remain following the prompt text, the "===>" character string is placed in the rightmost 4 positions of the prompt text space. When NONE is specified, no leader characters are added to the prompt text. When END is specified, the prompt text is right justified within the prompt text space.

If specified, this attribute overrides the PMTFMT value of the DTACOL tag.

PSVAR=point-and-shoot-variable | %varname
This attribute provides the name of a variable that is to be set when a DTAFLD is clicked on for point-and-shoot selection. You can define this attribute as a variable name preceded by a "%".

The *point-and-shoot-variable* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

PSVAL=point-and-shoot-value | %varname
This attribute provides the value to be placed in the field specified by the PSVAR attribute. You can define this attribute as a variable name preceded by a "%". To specify a blank value, the `"' '"` (quotation mark, apostrophe, blank, apostrophe, quotation mark) coding notation should be used.

PAS=%varname
This attribute can be used to provide a variable name to specify ON or OFF for point-and-shoot. When PSVAR and PSVAL have been specified without the PAS attribute, the point-and-shoot field will be automatically enabled.

CSRGRP=NO | YES | N
When CSRGRP=YES, the conversion utility generates a cursor group number to be used for this data field. When CSRGRP=n, the number provided is used for this field. The PAS attribute must be specified as %varname.

The CSRGRP attribute is accepted for all data fields. It is used at run-time for output fields only.

EXPAND
The EXPAND attribute, used in combination with EXPAND=xy on the PANEL definition, causes the expand characters to be added to the DTAFLD definition, effectively allowing ENTWIDTH to expand. The ENTWIDTH value must be specified as 4 or greater for the EXPAND function to operate.

Note: If the PANEL tag attribute EXPAND is defined with no value specified, the DTAFLD tag EXPAND attribute is not used.

FLDWIDTH=n
The FLDWIDTH attribute, used in combination with WINDOW=NO on the PANEL definition, provides the width of a DTAFLD which spans multiple lines.

FLDWIDTH cannot be used within any horizontal region.

ATTRCHANGE=NO | YES | NEW
When ATTRCHANGE=YES or ATTRCHANGE=NEW, the conversion utility formats an additional entry in the panel)ATTR section (that can apply to multiple data fields) instead of creating a unique ".ATTR(field-name)" entry in the)INIT section for this field. With this option, multiple DTAFLD tags with the same characteristics require fewer panel logic statements. ATTRCHANGE=NEW creates a new entry. ATTRCHANGE=YES uses an existing entry, if possible.

INIT=initial-value
When the INIT attribute is specified, the conversion utility adds a statement to the panel)INIT section to initialize the field to the *initial-value*.

DEPTH=n | %varname
This attribute defines the depth reserved for the field. When the panel is displayed in GUI mode, a field specified as point-and-shoot results in a push button displayed with the specified DEPTH. You use this attribute in

combination with the IMAPNAME attribute to provide space for the image. The minimum value is 1 and the maximum value is the remaining panel depth.

IMAPNAME=image-name | %varname
This attribute specifies the name of an image to be placed on the point-and-shoot push button when it is displayed in GUI mode. The *image-name* is not used when the panel is displayed in host mode.

The *image-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

IMAPNAMEP=image-namep | %varname
This attribute specifies the name of an image to be placed on the point-and-shoot push button after it has been pushed when it is displayed in GUI mode. The *image-namep* is not used when the panel is displayed in host mode.

The *image-namep* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

PLACE=ABOVE | BELOW | LEFT | RIGHT
This attribute specifies the position of the image relative to the text within the point-and-shoot push button.

DBALIGN=YES | NO | PROMPT | FIELD | FORCE
This attribute defines the DBALIGN value. DBALIGN is used only for DBCS language conversions when PMTLOC=ABOVE and the DBALIGN invocation option is specified.

When DBALIGN=PROMPT, the start position of the prompt-text is shifted 1 position to the right.

When DBALIGN=FIELD, the start position of the data field is shifted 1 position to the right.

When DBALIGN=YES, and the prompt-text starts with a DBCS character, the data field is shifted. If DBALIGN=YES and the prompt text starts with an SBCS character or the prompt text is not provided, no shifting is done.

When DBALIGN=FORCE, both the prompt-text and the data field are shifted. DBALIGN=YES and DBALIGN=FORCE are useful to align a DTAFLD with another DTAFLD or SELFLD tag.

When DBALIGN=NO, no alignment adjustment is made.

PMTSKIP=NO | YES
This attribute is used for horizontal formatting of input fields. When PMTSKIP=YES, and the previous DTAFLD definition includes the NOENDATTR attribute, the cursor moves past the prompt text to the input field when the user enters the last character in the previous field. If there is no other input field on the panel, the cursor returns to the first input field on the panel. The ISPF SKIP keyword is not supported in GUI mode.

DESSKIP=NO | YES
This attribute is used for horizontal formatting of input fields. When DESSKIP=YES, and the current DTAFLD definition includes the NOENDATTR attribute, the cursor skips over the description text provided by the DTAFLDD tag to the next input field when the user enters the last character in the current field. If there is no other input field on the panel, the cursor returns to the first input field on the panel. The ISPF SKIP keyword is not supported in GUI mode.

FLDTYPE=CUA | ISPF
This attribute defines the attribute type to be applied to the field. TYPE=CUA, the default, causes the field to display using the standard CUA attribute. When FLDTYPE=ISPF, a non-CUA attribute entry is generated for the)ATTR section, and you can specify the color, intensity, and highlighting of the attribute. See the COLOR, INTENS, and HILITE attributes that follow for more information. These attributes are not valid when FLDTYPE=CUA.

Note: IF DISPLAY=NO is specified, an .ATTR(...) is created to override this field.

COLOR=WHITE | RED | BLUE | GREEN | PINK | YELLOW | TURQ | %varname
This attribute specifies the color of the field. You can define this attribute as a variable name preceded by a percent (%) sign.

INTENS=HIGH | LOW | NON | %varname
This attribute defines the intensity of the field. You can define this attribute as a variable name preceded by a percent (%) sign.

HILITE=USCORE | BLINK | REVERSE | %varname
This attribute specifies the extended highlighting attribute of the field. You can define this attribute as a variable name preceded by a percent (%) sign.

ATTRCHAR=code
This attribute can be a single character or a two-position entry of valid hex digits. If you enter an incorrect value, a warning message is issued and the value is set to null. Hex entries are converted to character. Hex values '00'–'2F' are reserved for use by the conversion utility.

CAPS=OFF | ON
When CAPS=ON, the data in the field is displayed in uppercase characters.

NOJUMP=OFF | ON
When NOJUMP=ON, the JUMP function is disabled for the field.

AUTOTYPE=PROJECT | GROUP1 | GROUP2 | GROUP3 | GROUP4 | TYPE | MEMBER | DSN
This attribute specifies that ISPF panel logic be added to support the AUTOTYPE function.

AUTOTYPE=DSN is specified for data set name fields.

The other attribute values are used for ISPF- format project, group, type, and member name fields.

Multiple data fields can be specified with AUTOTYPE=DSN. Only one field can be specified with each of the other listed attribute values.

AUTOVOL = volser name
This attribute specifies an associated panel field for volume name when AUTOTYPE=DSN.

AUTODMEM = YES | NO
This attribute specifies whether a member name is part of the data set name when AUTOTYPE=DSN.

VARDCL = YES | NO
When VARDCL=NO the field name is not checked to the declared variable information provided with the VARCLASS and VARDCL tags.

prompt-text
This is the prompt text for the data field. The *prompt-text* appears in front of or

above the field, depending on the setting of the PMTLOC attribute. If you do not specify prompt text, no text appears for the field.

If the *prompt-text* exceeds the width defined for a prompt, it is word-wrapped to multiple lines.

Description

The DTAFLD tag defines an input field, an output field, or an input/output field on an application panel.

The formatted width of the field is 2 positions more than the ENTWIDTH value to provide for an attribute byte both before and after the field.

If PMTLOC=ABOVE, an attribute is placed both before and after the prompt text reserved space. If PMTLOC=BEFORE (or PMTLOC is not specified), and the DTAFLD is being formatted in a horizontal region, then an additional byte is used for the field prompt attribute when the field prompt is not at the left edge of the panel.

The DTAFLDD tag can be used to provide the description text for the data field.

Conditions

- You must code the DTAFLD tag within an AREA, DTACOL, PANEL, or REGION definition. See "AREA (Area)" on page 223, "DTACOL (Data Column)" on page 306, "PANEL (Panel)" on page 415, and "REGION (Region)" on page 445 for descriptions of these tags.
- The variable name specified in the DATAVAR attribute should have an associated VARDCL definition. See "VARDCL (Variable Declaration)" on page 486 for a complete description of this tag.
- If both PAD and PADC have been specified, PAD is ignored and PADC is used.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested Tags

You can code the following tags within a DTAFLD definition:

Tag	Name	Usage	Page	Required
ASSIGNL	Assignment list	Multiple	231	No
COMMENT	Comment	Multiple	282	No
DTAFLDD	Data field description	Multiple	323	No
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SOURCE	Source	Multiple	472	No

Example

The following source file markup contains an application panel with three data fields and the variable declarations and classes associated with the data fields. The **Date** field is an output-only field that displays the current date. The **Name** and **Password** fields are input/output fields. The **Password** field is defined as a

required field, and specifies DISPLAY=NO, so the user input for this field will not be displayed. A data column specifying a default prompt width for the data fields is also defined. Figure 117 shows the formatted result.

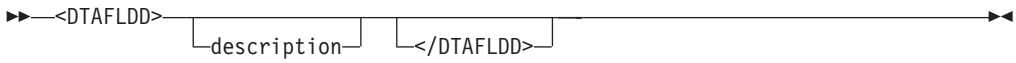
```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=date TYPE='char 8'>
<VARCLASS NAME=name TYPE='char 25'>
<VARCLASS NAME=password TYPE='char 8'>

<VARLIST>
  <VARDCL NAME=curdate VARCLASS=date>
  <VARDCL NAME=namevar VARCLASS=name>
  <VARDCL NAME=passvar VARCLASS=password>
</VARLIST>

<PANEL NAME=dtafld1 HELP=loghelp>System Logon
<TOPINST>Complete the following fields, then press Enter.
<AREA>
  <DTACOL PMTWIDTH=12>
    <DIVIDER>
    <DTAFLD DATAVAR=curdate USAGE=out ENTWIDTH=8 FLDSPACE=27>Date
  <DTAFLDD>(Current Date)
    <DIVIDER>
    <DTAFLD DATAVAR=namevar ENTWIDTH=25 DESWIDTH=25>Name
      <DTAFLDD>(Last, First)
    <DIVIDER>
    <DTAFLD DATAVAR=passvar REQUIRED=yes ENTWIDTH=8 DISPLAY=no>Password
  </DTACOL>
</AREA>
</PANEL>
```

```
                              System Logon

Complete the following fields, then press Enter.


Date . . . : 08/29/90                  (Current Date)

Name . . . . _____________________  (Last, First)

Password . .









 F1=Help    F3=Exit   F12=Cancel
```

Figure 117. Data Fields

DTAFLDD (Data Field Description)

The DTAFLDD tag defines descriptive text associated with a data field.



description
This is the descriptive text associated with the data field.

Description

The DTAFLDD tag defines descriptive text associated with a data field. For example, it could explain what the application user can type into the field.

The text appears in the area defined by the DESWIDTH attribute of the DTAFLD or DTACOL tag.

You can specify more than one DTAFLDD tag for a given field. Each data field description starts a new line.

Conditions

- You must code the DTAFLDD tag within the DTAFLD definition it is associated with. See "DTAFLD (Data Field)" on page 312 for a complete description of this tag.

Nested Tags

You can code the following tags within a DTAFLD definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

The following application panel markup contains two data fields that each have associated data field descriptions. Figure 118 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=filecls TYPE='char 8'>
<VARCLASS NAME=copycls TYPE='char 2'>

<VARLIST>
  <VARDCL NAME=file    VARCLASS=filecls>
  <VARDCL NAME=copynum VARCLASS=copycls>
</VARLIST>

<PANEL NAME=dtafldd>Print a File
<TOPINST>Type in the name of the file you want to print
and the number of copies, then press Enter.
<AREA>
  <DTAFLD DATAVAR=file PMTWIDTH=12 ENTWIDTH=8 DESWIDTH=30>Filename
    <DTAFLDD>(Maximum of 8 characters)
  <DTAFLD DATAVAR=copynum PMTWIDTH=12 ENTWIDTH=2 DESWIDTH=8>Copies
```

```
    <DTAFLDD>(1 - 99)
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

```
                                Print a File

Type in the name of the file you want to print and the number of copies,
then press Enter.

Filename . . ________  (Maximum of 8 characters)
Copies . . . __  (1 - 99)









Enter a command ===> _____________________________________________________
 F1=Help    F3=Exit   F12=Cancel
```

Figure 118. Data Field Descriptions

DTDIV (Definition Term Divider)

The DTDIV tag defines a visible vertical divider (|) between multiple DT tags.

Description

The DTDIV tag can be used to create a visual separation between the definition terms. Each DTDIV tag adds a vertical bar (plus display control attributes) to the Definition Term text.

Conditions

The DTDIV tag can be coded before the first DT tag, between DT tags, or following the last DT tag (before the DD tag definition).

Nested Tags

None.

Example

The following example illustrates the use of the DTDIV tag in combination with the multiple DT tag function and the DIVEND attribute of the DL tag. Figure 119 on page 326 shows the formatted result.

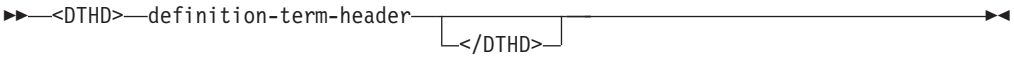
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=dtdiv DEPTH=22 WIDTH=66>Employee Code Help
  <AREA depth=1 extend=on>
  <INFO width=*>
    <P>The following list defines the valid employee codes.
    <DL TSIZE='14 4' BREAK=none COMPACT DIVEND=yes>
      <DLDIV TYPE=solid>
      <DTHDIV>
      <DTHD>Code
      <DTHDIV>
      <DTHD>Flag
      <DTHDIV>
      <DDHD>Meaning
      <DLDIV TYPE=solid>
      <DTDIV>
      <DT NOSKIP>Full-time
      <DTDIV>
      <DT FORMAT=center>F
      <DTDIV>
      <DD>Indicates that the employee works a
      regular schedule of 40 hours or more weekly.
      <DLDIV TYPE=solid>
      <DTDIV>
      <DT>Part-time
      <DTDIV>
      <DT FORMAT=center>P
      <DTDIV>
      <DD>Indicates that the employee works a regular
      schedule of 20 to 40 hours weekly.
      <DLDIV TYPE=solid>
      <DTDIV>
      <DT>Supplemental
      <DTDIV>
      <DT FORMAT=center>S
      <DTDIV>
      <DD>Indicates that the employee works less than
      20 hours weekly.
      No regular schedule is in place.
      <DLDIV TYPE=solid>
    </DL>
  </INFO>
  </AREA>
</HELP>
```

```
                    Employee Code Help

The following list defines the valid employee codes.
-------------------------------------------------------------
| Code           | Flag | Meaning                           |
-------------------------------------------------------------
| Full-time      |  F   | Indicates that the employee works a |
|                |      | regular schedule of 40 hours or more |
|                |      | weekly.                           |
-------------------------------------------------------------
| Part-time      |  P   | Indicates that the employee works a |
|                |      | regular schedule of 20 to 40 hours |
|                |      | weekly.                           |
-------------------------------------------------------------
| Supplemental   |  S   | Indicates that the employee works |
|                |      | less than 20 hours weekly. No     |
|                |      | regular schedule is in place.     |
-------------------------------------------------------------
```

Figure 119. Definition Term Divider

DTHD (Definition Term Header)

The DTHD tag defines the heading for the term column of a definition list.



definition-term-header
This is the text of the definition term header. The length of the text for the definition term header should be less than the specified TSIZE value in the DL tag. A warning message is issued if the length of the text exceeds the limit.

Description

The DTHD tag defines the heading for the term column of a definition list. You can code multiple DTHD tags within a definition list.

The conversion utility inserts a blank line between the header and the list items unless the COMPACT attribute is specified on the DL tag.

Conditions

- You must code the DTHD tag within a DL definition. See "DL (Definition List)" on page 298 for a complete description of this tag.
- Each DTHD tag must be paired with and precede a definition description header (DDHD) tag.

Nested Tags

You can code the following tags within a DTAFLD definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No

Tag	Name	Usage	Page	Required
RP	Reference phrase	Multiple	452	No

Example

The following help panel markup contains a definition term header with the text "Prefix". Figure 120 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=dthd DEPTH=18>Prefix Help
<AREA>
<INFO>
  <P>The following list defines each of the valid prefixes.
  <DL TSIZE=12>
    <DTHD>Prefix
    <DDHD>Meaning
    <DT>AU
    <DD>Automotive
    <DT>HB
    <DD>Health and beauty
    <DT>LG
    <DD>Lawn and garden
    <DT>SG
    <DD>Sporting goods
  </DL>
</INFO>
</AREA>
</HELP>
```

```
                  Prefix Help

The following list defines each of the valid
prefixes.

Prefix      Meaning

AU          Automotive

HB          Health and beauty

LG          Lawn and garden

SG          Sporting goods

 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 120. Definition Term Header

DTHDIV (Definition Term Header Divider)

The DTHDIV tag defines a visible vertical divider (|) between multiple DTHD tags.

Description

The DTHDIV tag can be used to create a visual separation between the definition term headings. Each DTHDIV tag adds a vertical bar (plus display control attributes) to the Definition Term Header text.

Conditions

The DTHDIV tag can be coded before the first DTHD tag, between DTHD tags, or following the last DTHD tag (before the DDHD tag definition).

Nested Tags

None.

Example

The following example illustrates the use of the DTHDIV tag in combination with the multiple DT tag function and the DIVEND attribute of the DL tag. Figure 121 on page 329 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=dthdiv DEPTH=22 WIDTH=66>Employee Code Help
  <AREA depth=1 extend=on>
  <INFO width=*>
    <P>The following list defines the valid employee codes.
    <DL TSIZE='14 4' BREAK=none COMPACT DIVEND=yes>
      <DLDIV TYPE=solid>
      <DTHDIV>
      <DTHD>Code
      <DTHDIV>
      <DTHD>Flag
      <DTHDIV>
      <DDHD>Meaning
      <DLDIV TYPE=solid>
      <DTDIV>
      <DT NOSKIP>Full-time
      <DTDIV>
      <DT FORMAT=center>F
      <DTDIV>
      <DD>Indicates that the employee works a
      regular schedule of 40 hours or more weekly.
      <DLDIV TYPE=solid>
      <DTDIV>
      <DT>Part-time
      <DTDIV>
      <DT FORMAT=center>P
      <DTDIV>
      <DD>Indicates that the employee works a regular
      schedule of 20 to 40 hours weekly.
      <DLDIV TYPE=solid>
      <DTDIV>
      <DT>Supplemental
```

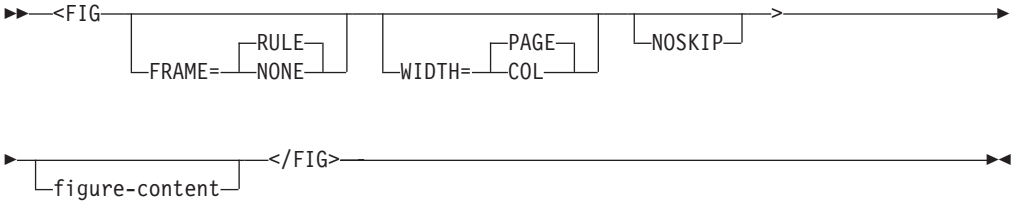
```
      <DTDIV>
      <DT FORMAT=center>S
      <DTDIV>
      <DD>Indicates that the employee works less than
      20 hours weekly.
      No regular schedule is in place.
      <DLDIV TYPE=solid>
    </DL>
  </INFO>
  </AREA>
</HELP>
```

```
                    Employee Code Help

The following list defines the valid employee codes.
-------------------------------------------------------------------
| Code           | Flag | Meaning                                 |
-------------------------------------------------------------------
| Full-time      |  F   | Indicates that the employee works a     |
|                |      | regular schedule of 40 hours or more    |
|                |      | weekly.                                 |
-------------------------------------------------------------------
| Part-time      |  P   | Indicates that the employee works a     |
|                |      | regular schedule of 20 to 40 hours      |
|                |      | weekly.                                 |
-------------------------------------------------------------------
| Supplemental   |  S   | Indicates that the employee works       |
|                |      | less than 20 hours weekly. No           |
|                |      | regular schedule is in place.           |
-------------------------------------------------------------------
```

Figure 121. Definition Term Header Divider

FIG (Figure)

The FIG tag defines the format of text so that it is set off from other text on the panel and retains the format of the enclosed text.



FRAME=RULE | NONE
This attribute specifies the type of frame to put around the figure.
RULE Specifies dashed lines appears above and below the figure.
NONE
Specifies no frame; a blank line is left above and below the figure.

WIDTH=PAGE | COL
This attribute specifies where the figure should be aligned. The value PAGE (the default) formats the figure on the original left margin. The value COL formats the figure on the current left margin. The current left margin may be

different than the original left margin of the panel if the FIG tag is nested within another tag that causes indenting; the UL tag, for example.

NOSKIP
This attribute causes the blank line normally placed before the figure to be skipped.

figure-content
This is the text of the figure definition.

Description

The FIG tag defines the format of text so that it is set off from other text on the panel and retains the format of the enclosed text. Tags that normally cause word wrapping within an information region (such as P, NOTE, or PARML) do not cause word-wrapping when nested within a FIG definition. In addition, blank spaces and blank lines in the source are preserved in the figure.

If any DTL source text line is too long to fit in the remaining available formatting width, the data is truncated. A warning message is issued when the first line within the figure is truncated.

A figure can also contain a figure caption, defined with the FIGCAP tag (see "FIGCAP (Figure Caption)" on page 331).

Conditions

- The FIG tag requires an end tag.
- You must code the FIG tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.

Nested Tags

You can code the following tags within a FIG definition:

Tag	Name	Usage	Page	Required
DL	Definition list	Multiple	298	No
FIGCAP	Figure caption	Single	331	No
HP	Highlighted phrase	Multiple	354	No
NOTE	Note	Multiple	400	No
NOTEL	Note List	Multiple	402	No
NT	Note	Multiple	404	No
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No
XMP	Example	Multiple	497	No

Example

The following help panel markup contains a figure definition with a ruled frame. The output of the text within the figure definition is identical to the *figure-content*. Figure 122 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=fig DEPTH=20>ShelfBrowse Help
<AREA>
<INFO>
  <FIG>

    We're your local library...

           CHECK US OUT!


  </FIG>
</INFO>
</AREA>
</HELP>
```

Figure 122. Figure

FIGCAP (Figure Caption)

The FIGCAP tag defines a caption for a figure defined with the FIG tag.

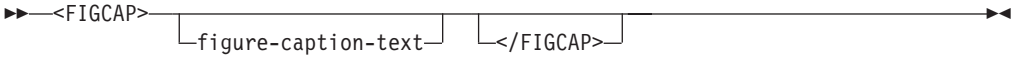


figure-caption-text
This is the text of the figure caption.

Description

The FIGCAP tag defines a caption for a figure defined with the FIG tag. The figure caption is formatted below the frame of the figure when FRAME=RULE is specified on the FIG tag.

The conversion utility does not add any blank lines before or after the figure caption.

Conditions

- You must code the FIGCAP tag within a FIG definition. See "FIG (Figure)" on page 329 for a complete description of this tag.
- You can code only one FIGCAP within a FIG definition. Code the FIGCAP tag following the content of the figure, before the FIG end tag.

Nested Tags

You can code the following tags within a FIG definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

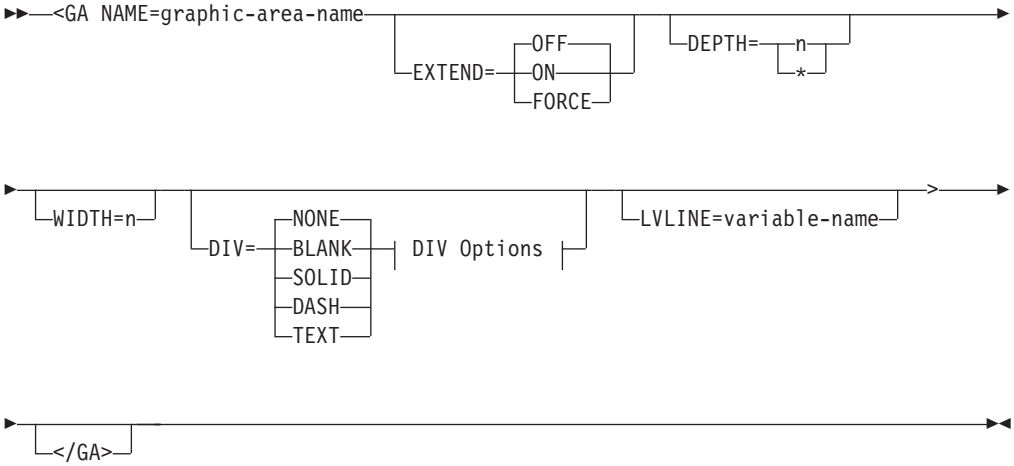
The following help panel markup contains a figure definition with an enclosed figure caption. Figure 123 on page 333 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=figcap DEPTH=20>ShelfBrowse Help
<AREA>
<INFO>
  <FIG>

    We're your local library...

           CHECK US OUT!


  <FIGCAP>Our Motto
  </FIG>
</INFO>
</AREA>
</HELP>
```

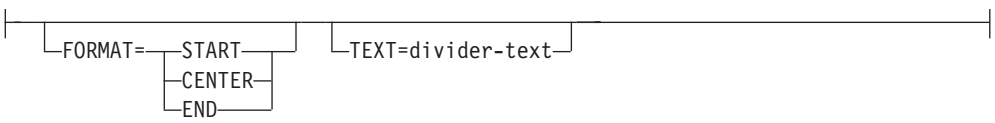
Figure 123. Figure Caption

GA (Graphic Area)

The GA tag allows the creation of graphic areas on ISPF panels.



DIV Options:



NAME=graphic-area-name
This attribute defines the name of the graphic area. This name is the dialog variable specified by the application that provides the data for the graphic area.

The NAME attribute must follow the standard naming convention described in "Rules for Variable Names" on page 211.

EXTEND=OFF | ON | FORCE
This attribute defines the run-time display size of the graphic area. If EXTEND=ON is specified, the graphic area definition is expanded to the size of the logical screen. If you intend to display the panel in a pop-up window, use EXTEND=OFF (which is the default.)

If EXTEND=FORCE is specified within a horizontal area or region, the EXTEND(ON) keyword is added to the graphic area attribute statement in the)ATTR panel section. The conversion utility issues a message to advise of a potential display error if other panel fields are formatted on or after the last defined line of the graphic area.

DEPTH=n | *
This attribute specifies the number of lines reserved for the graphic area definition. The DEPTH attribute value reserves space within the panel)BODY section. The minimum depth is one line. will reserve the remaining available panel depth for the graphic area.

WIDTH=n
This attribute specifies the number of columns reserved for the graphic area definition. The minimum width is the number of positions in the graphic area name plus 4 and the maximum is 2 positions less than the panel width. The conversion utility places attribute bytes on both sides of the graphic area.

DIV=NONE | BLANK | SOLID | DASH | TEXT
This attribute specifies the type of divider line to be placed before and/or after the graphic area. If this attribute is not specified or has the value NONE, no divider line is generated. The value BLANK produces a blank line. You must specify SOLID, DASH, or TEXT to produce a visible divider line. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line. A visible divider line formats with a non-displayable attribute byte on each end of the line.

FORMAT=START | CENTER | END
This attribute specifies the position of the *divider-text* within the divider line. You must specify both the FORMAT attribute and the TEXT attribute to create a divider line containing text.

TEXT=divider-text
This attribute specifies the text to be placed on the divider line. You must specify both the FORMAT attribute and the TEXT attribute to create a divider line containing text.

LVLINE=variable-name
This attribute allows you to specify the name of a variable which contains the result of the ISPF function LVLINE.

The LVLINE attribute must follow the standard naming convention described in "Rules for Variable Names" on page 211.

Description

The GA tag defines a graphic area in the panel)BODY section.

If you specify the CMDAREA tag within your DTL source file, it must appear before the GA tag when DEPTH=* is specified. The GA tag DEPTH may have to be

adjusted to allow for additional lines which result from tags present within the panel definition following the end GA tag.

Refer to the *ISPF User's Guide* for a discussion of the graphic area in ISPF panels.

Conditions

- You must code the GA tag within a PANEL, AREA, or REGION tag. If found anywhere else, an error is logged and the output panel is not saved.
- If NAME is not valid or not specified, an error is logged and the output panel is not saved.
- You can use the EXTEND=ON attribute only once within a panel. If EXTEND is already active, from another GA tag, or from an AREA, DA, SELFLD, or REGION tag, a warning message is logged and the EXTEND attribute is ignored.
- You can code only one GA tag within a PANEL definition.
- You cannot code the GA tag within a scrollable area.

Nested Tags

None.

Example

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampabc system>)>
&sampvar1;

<PANEL NAME=ga KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
<TOPINST> Type in patron's name and card number (if applicable)
<AREA>
  <DTACOL PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25 SELWIDTH=25>
    <DTAFLD DATAVAR=curdate USAGE=out ENTWIDTH=8>Date
    <DTAFLD DATAVAR=cardno ENTWIDTH=7>Card No.
      <DTAFLDD>(A 7-digit number)
    <DTAFLD DATAVAR=name>Name
      <DTAFLDD>(Last, First, M.I.)
    <DTAFLD DATAVAR=address>Address
  </DTACOL>
  <DIVIDER>
  <GA NAME=garea DIV=solid DEPTH=6 WIDTH=40>
  </GA>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

GENERATE (Generate)

The GENERATE tag provides direct formatting for)BODY and)AREA panel sections.

Description

The GENERATE tag is used to add pre-formatted displayable panel contents into the)BODY or)AREA panel sections. These contents can contain any valid displayable information. It is the panel developer's responsibility to provide valid displayable data.

The pre-formatted information is coded within a nested SOURCE tag. The SOURCE tag TYPE attribute is automatically determined based on the position of the GENERATE tag within the DTL source file. When panel attributes are required, the ATTR tag can be used to define the necessary)ATTR section entries.

Conditions

- The GENERATE tag requires an end tag.
- You must code the GENERATE tag within an AREA, HELP or PANEL tag definition.

Nested Tags

You can code the following tags within a FIG definition:

Tag	Name	Usage	Page	Required
ATTR	Attribute	Multiple	234	No
COMMENT	Comment	Multiple	282	No
SOURCE	Source	Multiple	472	No

Example

The following markup shows contains a GENERATE tag with nested ATTR and SOURCE tags. Figure 124 on page 337 shows the generated panel file.

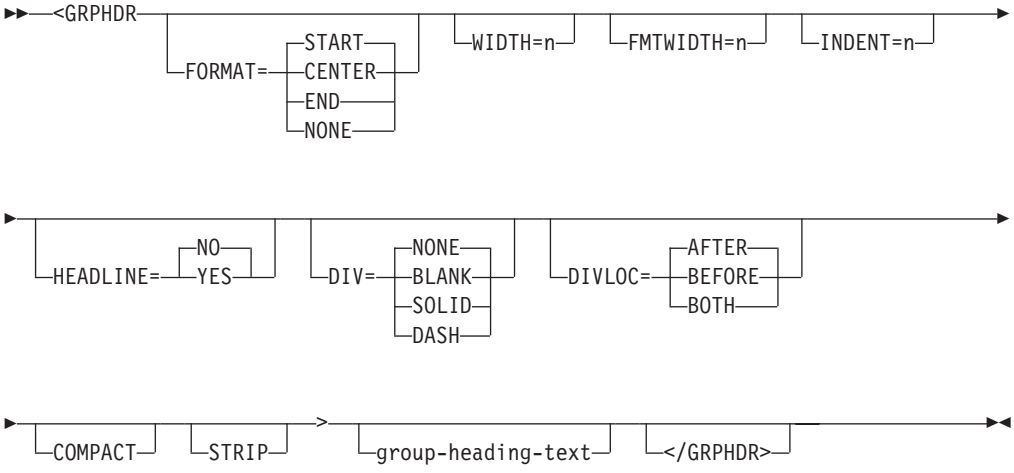
```
<:DOCTYPE DM SYSTEM>
<PANEL NAME=* KEYLIST=keylxmp applid=isr window=no>
            Generate Tag Example

<CMDAREA>
<pnlinst compact>
         Sample panel source to illustrate the GENERATE tag.

<divider type=solid gap=no>

<genereate>
    <attr attrchar=! type=FP>
    <attr attrchar=_ type=NEF>
    <attr attrchar=+ type=NT>

<source>
!  Project ===>_PROJECT !
!  Group   ===>_GROUP1  !===>_GROUP2  !===>_GROUP3  !===>_GROUP4  +
!  Type    ===>_TYPE    !
!  Member  ===>_MEMBER  !

!  DS Name ===>_OTHERDSN                                          +
!  Volume  ===>_VOLUME+
   </source>
</genereate>

</panel>
```

```
)PANEL KEYLIST(KEYLXMP,ISR)
)ATTR DEFAULT("""") FORMAT(MIX)
 05 TYPE(PT)
 06 TYPE(PIN)
 09 TYPE(FP)
 0A TYPE(NT)
 13 TYPE(NEF)
 22 TYPE(WASL) SKIP(ON) GE(ON)
 !  TYPE(FP)
 _  TYPE(NEF)
 +  TYPE(NT)
)BODY  CMD(ZCMD)
                              Generate Tag Example

 Command ===> Z


 Sample panel source to illustrate the GENERATE tag.
 ------------------------------------------------------------------------------
!  Project ===>_PROJECT !
!  Group   ===>_GROUP1  !===>_GROUP2  !===>_GROUP3  !===>_GROUP4  +
!  Type    ===>_TYPE    !
!  Member  ===>_MEMBER  !
!  DS Name ===>_OTHERDSN                                                    +
!  Volume  ===>_VOLUME+
)INIT
.ZVARS = '(ZCMD)'
&ZCMD = ' '
)PROC
)END
```

Figure 124. Generated Panel

GRPHDR (Group Header)

The GRPHDR tag allows the creation of group headers on ISPF panels.



FORMAT=START | CENTER | END | NONE
This attribute specifies the type of group header formatting.

When FORMAT=NONE, the lines of *group-heading-text* are placed in the panel)BODY section without alteration. The processing is similar to the LINES tag.

When the values START, CENTER, or END are specified, the data is processed in a manner similar to the P tag. The *group-heading-text* is read and flowed to fit within the width limit specified by FMTWIDTH. Multiple lines may be added to the panel, depending on the length of the *group-heading-text*.

WIDTH=n
This attribute specifies the number of columns reserved for the group heading. The minimum width for a group heading is 4. The maximum value is the remaining panel width. If WIDTH is not specified, the default value is set to the remaining panel width. The conversion utility uses 2 positions from the specified or default WIDTH for attributes.

FMTWIDTH=n
This attribute specifies the number of columns to use for formatting the *group-heading-text*. The minimum formatting width is 2. The maximum value is the value specified or defaulted for WIDTH. If FMTWIDTH is not specified, the default value is set to the value of WIDTH.

INDENT=n
This attribute specifies that the group heading is to be indented from the current position.

HEADLINE=NO | YES
This attribute specifies whether dashes are added to span the width of the group heading not occupied by text. This allows a visual indication of the width of the group heading.

DIV=NONE | BLANK | SOLID | DASH
This attribute specifies the type of divider line to be placed before and after the group heading. If this attribute is not specified or has the value NONE, no divider line is generated. The value BLANK produces a blank line. You must specify SOLID or DASH to produce a visible divider line. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line.

DIVLOC=AFTER | BEFORE | BOTH
This attribute specifies whether divider line is to be added after the group heading, before the group heading or both before and after the group heading.

COMPACT
This attribute causes the group heading to format without a blank before the heading.

STRIP
This attribute causes leading and trailing blanks to be removed from the heading.

group-heading-text
This is the text of the group header. If no *group-heading-text* is provided, a blank line is added to the panel unless the COMPACT attribute is also specified.

Description

The GRPHDR tag defines a group heading in the panel)BODY section.

The FMTWIDTH and HEADLINE attributes are not valid in combination with FORMAT=NONE. The DIVLOC attribute is not valid in combination with DIV=NONE.

You use the FMTWIDTH attribute to control the width of flowed text within the number of columns specified by WIDTH. The FORMAT attribute controls the placement of the resulting lines within the heading WIDTH. The FMTWIDTH attribute has no effect if the length of the *group-heading-text* is less than the value specified.

Because the group heading is formatted as text, a blank line is placed at the beginning of each group heading unless the COMPACT attribute has been specified. However, when the group heading is the first item in a scrollable region the blank line is not generated.

Conditions

- You must code the GRPHDR tag within a PANEL, AREA, DTACOL, or REGION tag. If found anywhere else, an error is logged and the output panel is not saved.

Nested Tags

You can code the following tag within a GRPHDR tag:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

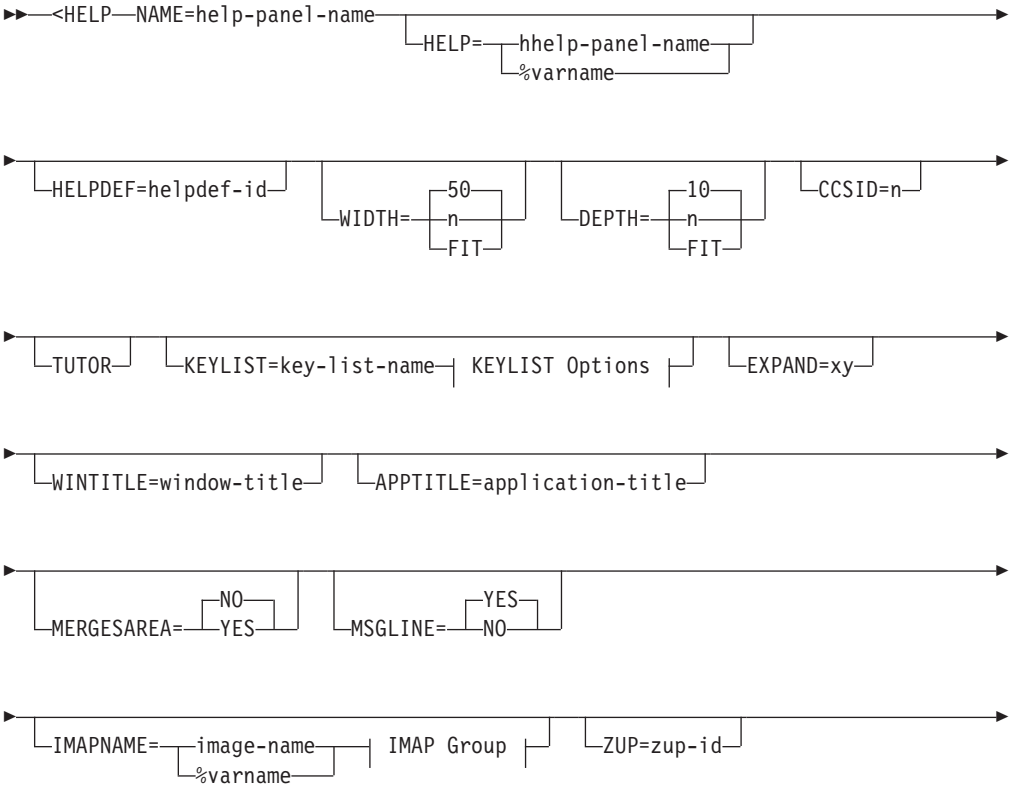
```
<!DOCTYPE DM SYSTEM>
  <!entity sampvar1 system>
  <!entity sampabc system>)>
&sampvar1;

<PANEL NAME=grphdr KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
<TOPINST> Type in patron's name and card number (if applicable)
<AREA>
  <GRPHDR FORMAT=center WIDTH=50 FMTWIDTH=30 DIV=solid COMPACT>
        Data Field Group Heading
  </GRPHDR>
  <DTACOL PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25 SELWIDTH=25>
    <DTAFLD DATAVAR=curdate USAGE=out ENTWIDTH=8>Date
    <DTAFLD DATAVAR=cardno ENTWIDTH=7>Card No.
      <DTAFLDD>(A 7-digit number)
    <DTAFLD DATAVAR=name>Name
      <DTAFLDD>(Last, First, M.I.)
    <DTAFLD DATAVAR=address>Address
  </DTACOL>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

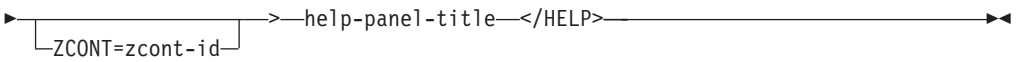
```
  File  Search  Help
------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number (if applicable)

            Data Field Group Heading
---------------------------------------------
Date . . . : ________
Card No. . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________




Enter a command ===> ____________________________________________________________
 F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

Figure 125. Group Heading

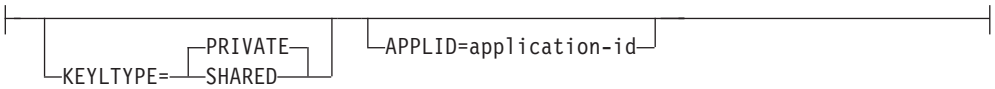
HELP (Help Panel)

The HELP tag defines a help panel.

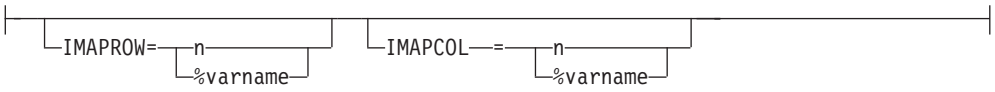




KEYLIST Options:



IMAP Group:



NAME=help-panel-name

This attribute specifies the name of the help panel. The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

In addition, the *help-panel-name* is limited to 7 characters when the DTL source you are converting causes the conversion utility to build multiple panels. If you have specified an 8-position help name and multiple panels are required, the help name will be truncated to 7 positions. If you are not creating a scrollable help panel, this allows additional panels to be built if the help text exceeds the limits of the original help panel. Up to 36 additional help panels will be built to contain additional help text.

If the number of generated panels required exceeds 37, a warning message is issued and all help text after the 37th panel is discarded. The additional panel names are generated from the original *help-panel-name* by the following rules:

- The character 'X' will be used to pad the *help-panel-name* to 8 characters in length if the original *help-panel-name* is less than 8 characters.
- The eighth character of the generated panel name will be incremented from 0–9 and A–Z depending on the number of panels required to be generated. For example, if the original *help-panel-name* is 'HELP1' and the help text extends beyond the original panel, the second generated panel name would be 'HELP1XX0', and the third would be 'HELP1XX1'.

If you specify NAME=*, the *help-panel-name* is set to the input DTL source member name. If multiple dialog element definitions have been combined within a single source file, then this notation should be used for only one dialog element definition within the file. See "Dialog Elements" on page 5 for a description of dialog element types created by the conversion utility.

The *help-panel-name* is used to build the help panel output file name in which the conversion utility stores the converted help panel. The default name is "userid.PANELS(*help-panel-name*)".

The output panel filename can be specified on the invocation panel for the conversion utility. You can specify the panel library of your choice. If the SCRIPT option was specified, the *help-panel-name* is also used to build the file

name in which the conversion utility stores the image of the help panel. The default name is "userid.SCRIPT(*help-panel-name*)".

See "Chapter 10. Using the Conversion Utility" on page 175 for complete information on invocation syntax.

The ISPF tutorial facility displays help panels. The user can scroll forward by pressing Enter or the RIGHT (F11) key, or scroll backward by pressing the LEFT (F10) key. The scrolling indicators "**More: +**", "**More: –**", and "**More: – +**" are added to the displayed panel to indicate more help is available.

HELP=hhelp-panel-name | %varname
This attribute specifies the name of a defined help for help panel. It identifies the help text that is associated with help processing.

The *hhelp-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

Specification of the HELP attribute will cause ISPDTLC to generate ".HHELP=*hhelp-panel-name*" (or ".HHELP=&varname") in the)INIT section during help panel generation.

If no value is provided for the HELP attribute, the conversion utility will add the default ".HHELP = ISP00006" to the generated panel.

ISPF displays this panel when the application user requests help and the cursor is not on a panel field that is defined as a reference phrase.

HELPDEF=helpdef-id
This attribute specifies a defined help default. The *helpdef-id* value is the identifier specified on the HELPDEF tag. You can override any of the defaults from this HELPDEF tag by specifying that attribute on the HELP tag. See the description of the HELPDEF tag for information on defining help defaults.

WIDTH=50 | n | FIT
This attribute specifies the width of the help panel. The default width is 50. When you specify this attribute, it should be greater than or equal to the minimum width of 16 characters. The maximum is 156. Because there are set margins of 1 character on each side of the panel text to allow for 3270 attribute bytes, the effective width for text for a help panel defined as WIDTH=50 is 48 characters.

If you have specified WIDTH=FIT, The conversion utility will format the panel using the maximum available width. When formatting is completed the WIDTH value will be reset to the minimum width used or to 16 if the formatted panel is less than 16 characters wide.

If the specified WIDTH exceeds the maximum minus 4 allowed by the display device, ISPF issues an error message at run time.

DEPTH=10 | n | FIT
This attribute specifies the depth of the HELP panel. The maximum depth is 60 and the minimum depth is 6. When the panel body does not end with a scrollable area, four lines at the bottom of each help panel are reserved for the function key area. Two lines are reserved at the top of the help panel for the *help-panel-title* and a separator line. You must include provisions for these 6 lines in the depth you specify.

The default help panel depth of 10 is used when the DEPTH attribute provided cannot be used or the DEPTH attribute is not specified.

If you have specified DEPTH=FIT, The conversion utility will format the panel using a depth of 22. When formatting is completed the DEPTH value will be reset to the minimum depth used or to 6 if the formatted panel contains less than 6 lines.

If the specified DEPTH exceeds the maximum, minus 2, allowed by the display device, ISPF issues an error message at run time.

CCSID=n
CCSID specifies the coded-character-set identifier as defined by the Character Data Representation Architecture. CCSID should be entered as a five-position numeric value. For more information on using the CCSID attribute, refer to the *ISPF Dialog Developer's Guide and Reference*

TUTOR
This attribute specifies that the panel title be formatted with the word ***Tutorial*** (or its translated equivalent) on each end of the title line, similar to ISPF tutorial panels.

KEYLIST=key-list-name
KEYLIST is an ISPF extension to the Dialog Tag Language. This attribute specifies the name of the key mapping list associated with the help panel. If you do not specify a *key-list-name* in a HELP definition, the ISPF-provided key list (ISPHELP) will be used. For information on defining key mapping list, see "KEYL (Key List)" on page 361. For information on the ISPF-provided key list, refer to the *ISPF User's Guide*

KEYLTYPE= PRIVATE | SHARED
This attribute is used to add the SHARED keyword to the KEYLIST parameter of the)PANEL statement. For information about the)PANEL statement, refer to the *ISPF Dialog Developer's Guide and Reference* The KEYLTYPE attribute is ignored if you have not provided the KEYLIST attribute as part of the HELP tag definition or as part of an associated HELPDEF tag definition.

APPLID=application-id
This attribute is used to add the application ID to the)PANEL statement. The *application-id* overrides the KEYLAPPL invocation option value. The APPLID attribute is ignored if you have not provided the KEYLIST attribute as part of the HELP tag definition or as part of an associated HELPDEF tag definition.

EXPAND=xy
This attribute adds the EXPAND(xy) attribute to the)BODY section of the panel. If only one character is present, the second character will be set to the same value. If the EXPAND attribute is present with no value specified, the conversion utility will use a character from the range of low-order hex values available for panel attributes. This removes an available character from possible use as a panel attribute and may cause panel formatting errors.

WINTITLE=window-title
This attribute is used to add a title on the pop-up window border. The attribute value is placed in the ISPF ZWINTTL variable. The maximum length of the *window-title* text is the panel width minus 1.

APPTITLE=application-title
This attribute is used to add a title on the GUI window border. The attribute value is placed in the ISPF ZAPPTTL variable. The maximum length of the *application-title* text is the panel width minus 1.

MERGESAREA= NO | YES
This attribute controls an additional formatting step for panels with a single scrollable area. If the entire contents of the scrollable area will fit within a standard 24-line panel (allowing 4 lines for the function keys display), the scrollable area content is moved into the panel body.

MSGLINE=YES | NO
This attribute controls the provision for a long message line in the generated panel. When MSGLINE=NO, the blank line for the long message is not added to the panel)BODY section. It is the panel designer's responsibility to ensure that critical panel areas are positioned so that the long message will not inhibit use of the resulting panel.

IMAPNAME=image-name | %varname
This attribute specifies the name of an image to be placed on the panel when it is displayed in GUI mode. The *image-name* is not used when the panel is displayed in host mode.

The *image-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

IMAPROW=n | %varname
This attribute specifies the row number for positioning the image. Image position uses an origin based on 0. Therefore, the minimum row value is 0 and the maximum is 59, relating to the above description for the DEPTH attribute. If a variable name is used, the application must set the variable to a valid value before the panel is displayed. The value specified should be within the actual panel depth for the image to be visible when the panel is displayed.

IMAPCOL=n | %varname
This attribute specifies the column number for positioning the image. Image position uses an origin based on 0. Therefore, the minimum column value is 0 and the maximum is 155, relating to the above description for the WIDTH attribute. If a variable name is used, the application must set the variable to a valid value before the panel is displayed. The value specified should be within the actual panel width for the image to be visible when the panel is displayed.

ZUP=zup-id
This attribute provides the name of the Tutorial panel to be assigned to the ZUP variable. It is valid only when the TUTOR attribute has also been specified.

ZCONT=zcont-id
This attribute provides the name of the Tutorial panel to be assigned to the ZCONT variable. It is valid only when the TUTOR attribute has also been specified.

help-panel-title
This specifies the title that appears on the help panel.

The *help-panel-title* is centered within the specified help panel width in accordance to CUA rules. If the title text is wider than the WIDTH specified, the title will be truncated with an ellipses (...) appended. Two lines are reserved for the title and a separator which can include the scrolling indicator if there are more panels.

Description

The HELP tag defines a help panel. A help panel can contain multiple information areas, which you use the INFO tag to define (see "INFO (Information Region)" on page 356).

ISPF always displays help panels defined with DTL in a pop-up window with a border. Therefore, the maximum value you can specify for the WIDTH attribute is 4 less than the maximum allowed by the display device. This allows for the left and right borders and their 3270 attribute characters. The maximum value for the DEPTH attribute is 2 less than the maximum allowed by the display device to allow for the top and bottom borders. Borders are added to the formatted help panel at run time.

If you are not creating a scrollable help panel and the text to be included in the)BODY section of the ISPF panel exceeds the specified DEPTH value, up to 36 additional panels are generated to contain the additional text. If the help text extends beyond the original help panel and 36 additional help panels, an error message will be issued and the excess text will be truncated. If the error occurs, and the DEPTH and WIDTH attributes are not set to their maximum values, the values should be increased or the amount of text to be included in the help panel should be reduced.

For nonscrollable HELP panels or for scrollable HELP panels which end with a nonscrollable section, a function key area of four lines is reserved at the bottom of the panel. The four lines are taken from the value specified for the DEPTH attribute.

If you do not specify the KEYLIST attribute, ISPF automatically associates the ISPF-provided key list "ISPHELP" with all DTL help panels.

The following table shows the "ISPHELP" key list and assignments:

Table 4. ISPHELP keylist and assignments

Key	Command	Key Label	Format
F1	HELP	Help	Short
F2	SPLIT	Split	Long
F3	EXIT	Exit	Short
F4	RESIZE	Resize	Long
F5	EXHELP	Exhelp	Short
F6	KEYSHELP	Keyshelp	Short
F7	UP	PrvTopic	Short
F8	DOWN	NxtTopic	Short
F9	SWAP	Swap	Long
F10	LEFT	PrvPage	Short
F11	RIGHT	NxtPage	Short
F12	CANCEL	Cancel	Short

All ISPHELP function keys are active when the cursor is in the help panel. Display of keys in the function key area is controlled by the user through the ISPF FKA command.

Because help panels are displayed by the ISPF tutorial processor, the commands assigned to the keys are those supported by the ISPF tutorial. For more information on the ISPF tutorial, refer to the *ISPF User's Guide*

Since ISPDTLC generated panels are not normally used in a full Tutorial, the default ISPHELP keylist may result in confusion in the use of the F7 and F8 keys for scrolling. An alternate approach is the ISPHLP2 keylist as defined below. To use this keylist, add the KEYLIST=ISPHLP2 attribute to your help panel definition.

Table 5. ISPHLP2 keylist and assignments

Key	Command	Key Label	Format
F1	HELP	Help	Short
F2	SPLIT	Split	Long
F3	EXIT	Exit	Short
F4	RESIZE	Resize	Long
F5	EXHELP	Exhelp	Short
F6	KEYSHELP	Keyshelp	Short
F7	LEFT	PrvPage	Short
F8	RIGHT	NxtPage	Short
F9	SWAP	Swap	Long
F10	LEFT	PrvPage	Long
F11	RIGHT	NxtPage	Long
F12	CANCEL	Cancel	Short

Conditions

- The HELP tag requires an end tag.
- You cannot code the HELP tag within any other tag definition.
- If the help panel does not have a panel body, the conversion utility issues an error message. The help panel must contain at least one INFO (information region) definition to qualify as a panel body. See "INFO (Information Region)" on page 356 for a complete description of this tag.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested Tags

You can code the following tags within a HELP definition:

Tag	Name	Usage	Page	Required
AREA	Area	Multiple	223	No
COMMENT	Comment	Multiple	282	No
DIVIDER	Divider	Multiple	296	No
GENERATE	Generate	Multiple	335	No
HP	Highlighted phrase	Multiple	354	Yes
INFO	Information region	Multiple	356	Yes
REGION	Region	Multiple	445	No

Tag	Name	Usage	Page	Required
SOURCE	Source	Multiple	472	No
TEXTLINE	Text Line	Single	475	No

Example

The following help panel markup contains an information region that contains a paragraph, a definition list, and two unordered lists nested within the definition list. Because all of the data does not fit in one help panel, the conversion utility created three panels HELP, HELPXXX0, and HELPXXX1. The panels are scrollable. Figures 126, 127, and 128 show the formatted results with the function key area displayed in its short form.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=help WIDTH=46 DEPTH=16>ShelfBrowse for Kids
<AREA>
  <INFO>
    <P>ShelfBrowse can help you
    find any kind of book you are looking for.
    The two main categories for books are:
    <DL TSIZE=12>
      <DTHD>Book
      <DDHD>Description
      <DT>Fiction
      <DD>Fiction books are stories
      that never really happened.
      The writer made them up.
      For example:
        <UL>
          <LI>Fairy Tales
          <LI>Mysteries
          <LI>Science fiction stories
        </UL>
      <DT>Nonfiction
      <DD>Nonfiction books are about
      things that really exist.
      For example:
        <UL>
          <LI>History books
          <LI>Reference books
          <LI>How to books
        </UL>
    </DL>
  </INFO>
</AREA>
</HELP>
```

```
            ShelfBrowse for Kids
                                More:     +
ShelfBrowse can help you find any kind of
book you are looking for. The two main
categories for books are:

Book        Description

Fiction     Fiction books are stories that
            never really happened. The
            writer made them up. For
            example:

 F1=Help        F3=Exit        F5=Exhelp
 F6=Keyshelp    F7=PrvTopic    F8=NxtTopic
F10=PrvPage    F11=NxtPage    F12=Cancel
```

Figure 126. Help Panel (Example 1 of 3)

```
            ShelfBrowse for Kids
                                More:   - +
            o   Fairy Tales

            o   Mysteries

            o   Science fiction stories

Nonfiction  Nonfiction books are about
            things that really exist. For
            example:


 F1=Help        F3=Exit        F5=Exhelp
 F6=Keyshelp    F7=PrvTopic    F8=NxtTopic
F10=PrvPage    F11=NxtPage    F12=Cancel
```

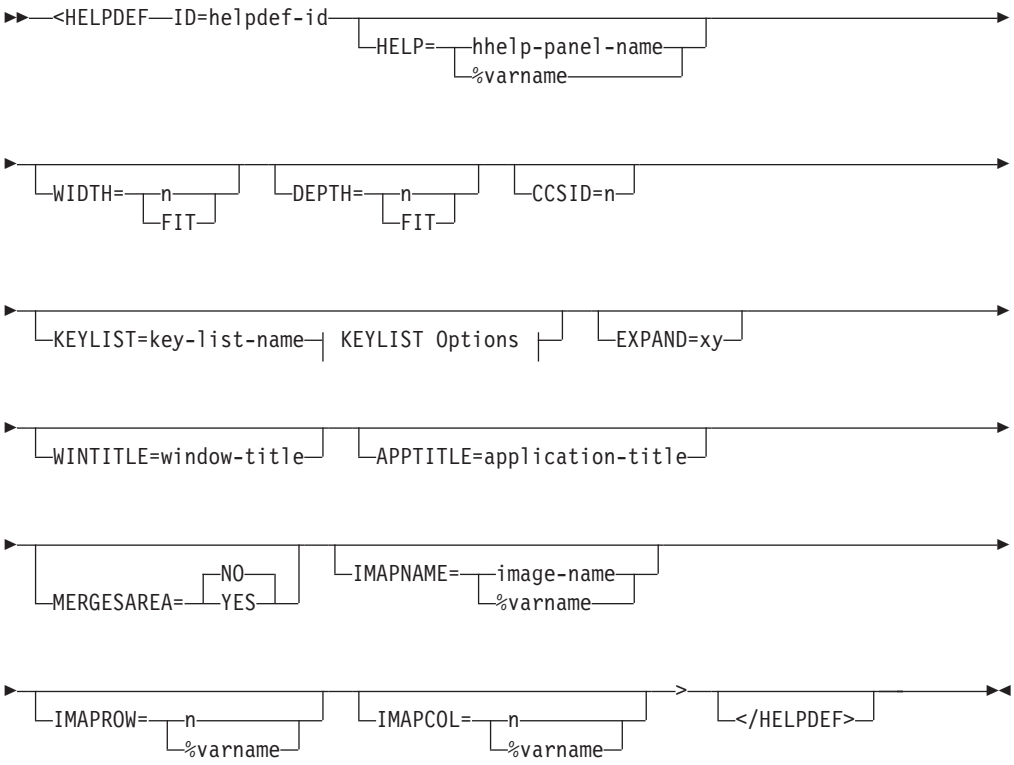
Figure 127. Help Panel (Example 2 of 3)

```
            ShelfBrowse for Kids
                                More:   -
            o   History books

            o   Reference books

            o   How to books




 F1=Help        F3=Exit        F5=Exhelp
 F6=Keyshelp    F7=PrvTopic    F8=NxtTopic
F10=PrvPage    F11=NxtPage    F12=Cancel
```

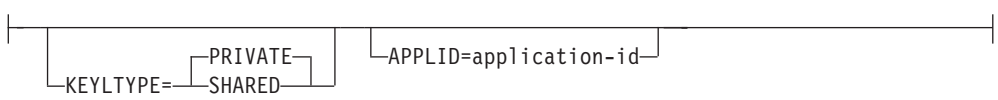
Figure 128. Help Panel (Example 3 of 3)

HELPDEF (Help default)

The HELPDEF tag defines default values for help panels.



KEYLIST Options:



ID=helpdef-id
This is the ID of the help panel default definition. The ID is used as the identifier of this set of default definitions on the HELP tag.

The *helpdef-id* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

HELP=hhelp-panel-name | %varname
This attribute specifies the default name of a defined help for help panel. It identifies the help text that is associated with help processing.

The *hhelp-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

WIDTH=n | FIT
This attribute specifies a default width value for a help panel that refers to this help default.

DEPTH=n | FIT
This attribute specifies a default depth value for a help panel that refers to this help default.

CCSID=n
CCSID specifies the coded-character-set identifier as defined by the Character Data Representation Architecture. CCSID should be entered as a five-position numeric value. For more information on using the CCSID attribute, refer to the *ISPF Dialog Developer's Guide and Reference*

KEYLIST=key-list-name
This attribute specifies the name of the key mapping list associated with the help panel. If you do not specify a *key-list-name* in a HELP definition, the ISPF-provided key list (ISPHELP) will be used. For information on defining key mapping list, see "KEYL (Key List)" on page 361. For information on the ISPF-provided key list, refer to the *ISPF User's Guide*

KEYLTYPE=PRIVATE | SHARED
This attribute is used to add the SHARED keyword to the KEYLIST parameter of the)PANEL statement. For information about the)PANEL statement, refer to the *ISPF Dialog Developer's Guide and Reference*

APPLID=application-id
This attribute is used to add the application ID to the)PANEL statement. The *application-id* overrides the KEYLAPPL invocation option value.

EXPAND=xy
This attribute adds the EXPAND(xy) attribute to the)BODY section of the panel. If only one character is provided, the second character is set to the same value. If the EXPAND attribute is present with no value specified, the conversion utility will use a character from the range of low-order hex values available for panel attributes. This removes an available character from possible use as a panel attribute and may cause panel formatting errors.

WINTITLE=window-title
This attribute is used to add a title on the pop-up window border. The attribute value is placed in the ISPF ZWINTTL variable. The maximum length of the *window-title* is the panel width minus 1.

APPTITLE=application-title
This attribute is used to add a title on the GUI window border. The attribute value is placed in the ISPF ZAPPTTL variable. The maximum length of the *application-title* text is the panel width minus 1.

MERGESAREA= NO | YES
This attribute controls an additional formatting step for panels with a single scrollable area. If the entire contents of the scrollable area will fit within a standard 24-line panel (allowing 4 lines for the function keys display), the scrollable area content is moved into the panel body.

IMAPNAME=image-name | %varname
This attribute specifies the name of an image to be placed on the panel when it is displayed in GUI mode. The *image-name* is not used when the panel is displayed in host mode.

The *image-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

IMAPROW=n | %varname
This attribute specifies the row number for positioning the image. Image position uses an origin based on 0. Therefore, the minimum row value is 0 and the maximum is 59, relating to the description for the DEPTH attribute on the HELP tag. If a variable name is used, the application must set the variable to a valid value before the panel is displayed. The value specified should be within the actual panel depth for the image to be visible when the panel is displayed.

IMAPCOL=n | %varname
This attribute specifies the column number for positioning the image. Image position uses an origin based on 0. Therefore, the minimum column value is 0 and the maximum is 155, relating to the description for the WIDTH attribute on the HELP tag. If a variable name is used, the application must set the variable to a valid value before the panel is displayed. The value specified should be within the actual panel width for the image to be visible when the panel is displayed.

Description

The HELPDEF tag defines default values for help panels. When a HELP panel tag refers to a help panel default, the values specified by the associated HELPDEF tag are used for the help panel unless overridden by values specified in the HELP tag definition.

The HELP tag can override any of the HELPDEF values by specifying that value within its own definition. Therefore, it is possible for a HELP tag to select certain default values from the help panel default and override others.

See "HELP (Help Panel)" on page 340 for more information.

You can code multiple HELPDEF definitions in a single application. Each help default must have a unique *helpdef-id*.

Portability Considerations

In ISPF Version 3 Release 1, the Conversion Utility supported the HELPDEF tag. In this release, the HELPDEF tag is supported to ease portability between releases.

Conditions

- You cannot code the HELPDEF tag within any other tag definition.
- You must code the HELPDEF tag before you code any HELP tag that refers to it.

Nested Tags

None.

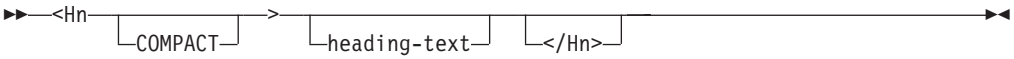
Example

In the following source file example, the HELPDEF definition defines default DEPTH and WIDTH values. The help panels “help15” and “help16” both reference the help default–“help15” uses both default values and “help16” uses only the default WIDTH value, and overrides the default DEPTH value by specifying its own DEPTH value. The help panel “help17” does not reference the help default, and defines its own DEPTH and WIDTH values.

```
<!DOCTYPE DM SYSTEM>

<HELPDEF ID=helpdef1 DEPTH=10 WIDTH=40>

<HELP NAME=help15 HELPDEF=helpdef1>Help for This
⋮

</HELP>

<HELP NAME=help16 HELPDEF=helpdef1 DEPTH=15>Help for That
⋮

</HELP>

<HELP NAME=help17 DEPTH=15 WIDTH=25>Help for the Other
⋮

</HELP>
```

Hn (Heading)

The heading tags define main topics and subtopics of information within an information region.



COMPACT
This attribute causes the heading-text to be formatted without creating a blank line before the heading.

heading-text
This is the text of the heading.

Description

The heading tags define main topics and subtopics of information within an information region. You can define up to four heading levels. The *n* in Hn indicates the heading level. The heading levels are formatted as follows:

H1 Identifies a main topic of information. The text is centered on the panel.

H2, H3, H4
The text is formatted against the left margin of the panel body.

Headings are formatted with one blank line before them.

Conditions

- The Hn tag must be coded within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.

Nested Tags

You can code the following tags within an H2, H3, and H4 tag only:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

The following help panel markup contains two levels of headings. Figure 129 on page 354 shows the formatted result.

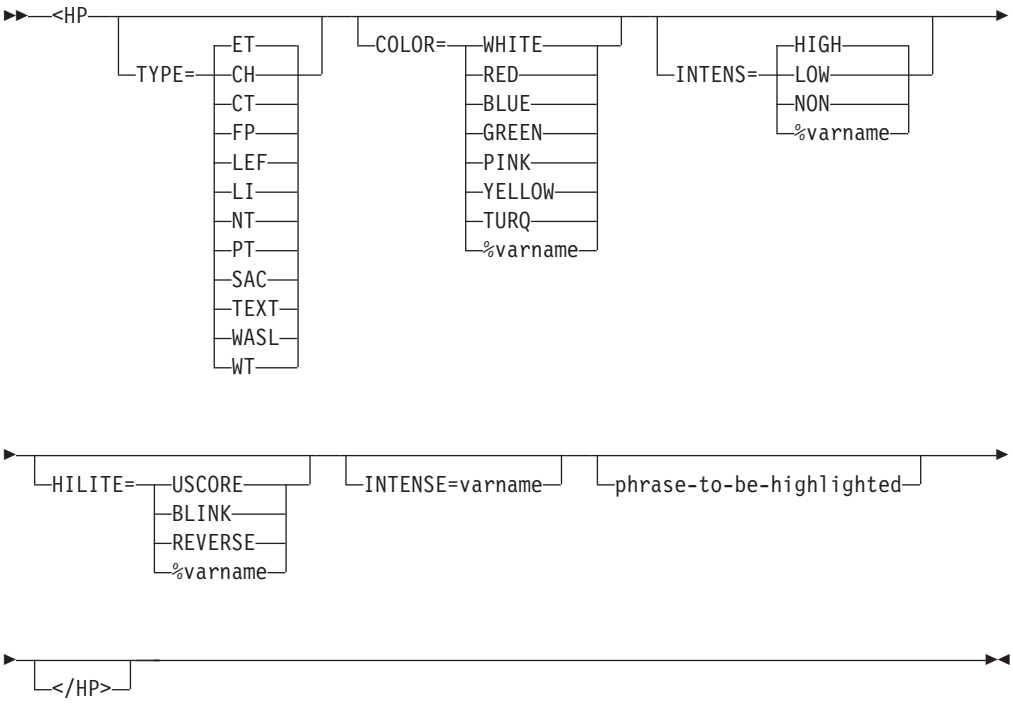
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=hn DEPTH=22>Department Descriptions Help
<AREA>
<INFO>
  <H1>Departments
  <H2>Entertainment
  <P>Our entertainment department carries the
  finest in home entertainment components.
  <H2>Exotic Pets
  <P>You can order from a wide variety of exotic
  pets and pet supplies in this department.
  <H2>Toys
  <P>Your kids will love our wide selection of
  toys, games, and play equipment.
</INFO>
</AREA>
</HELP>
```

```
          Department Descriptions Help


                   Departments

Entertainment

Our entertainment department carries the finest
in home entertainment components.

Exotic Pets

You can order from a wide variety of exotic pets
and pet supplies in this department.

Toys

Your kids will love our wide selection of toys,
games, and play equipment.

 F1=Help        F3=Exit        F5=Exhelp
 F6=Keyshelp    F7=PrvTopic    F8=NxtTopic
F10=PrvPage    F11=NxtPage    F12=Cancel
```

Figure 129. Headings

HP (Highlighted Phrase)

The HP tag identifies text to be displayed with highlighted emphasis.



TYPE= ET | CH | CT | FP | LEF | LI | NT | PT | SAC | TEXT | WASL | WT
This attribute defines the attribute type to be applied to the *phrase-to-be-highlighted*. Using a CUA attribute causes the text to appear in the associated color.

When TYPE=TEXT, a non-CUA attribute is generated and you can specify the color, intensity, and highlighting with the COLOR, INTENS, and HILITE attributes. These attributes are not valid for CUA types.

COLOR= WHITE | RED | BLUE | GREEN | PINK | YELLOW | TURQ | %varname
This attribute specifies the color of the field. You can define this attribute as a variable name preceded by a percent (%) sign.

INTENS= HIGH | LOW | NON | %varname
This attribute defines the intensity of a field. You can define this attribute as a variable name preceded by a percent (%) sign.

HILITE= USCORE | BLINK | REVERSE | %varname
This attribute specifies the extended highlighting attribute of a field. You can define this attribute as a variable name preceded by a percent (%) sign.

INTENSE=varname
This attribute supplies a variable name that must contain a valid value for the INTENS keyword. The entire phrase is controlled by this value. For example, if the variable contains the value NON, the phrase will not be visible.

phrase-to-be-highlighted
This text displays with highlighted emphasis.

Description

The HP identifies text to be displayed with highlighted emphasis by ISPF. The HP end tag restores normal text.

Conditions

- You can code the HP tag wherever the RP tag is valid.
- You can code the HP tag within the text following the CHDIV, CMDAREA, HELP, and PANEL tags.
- The HP tag requires an end tag.

Nested Tags

None.

Example

The following markup shows the formatted result in Figure 130 on page 356.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampabc system>)>
&sampvar1;

<PANEL NAME=hp KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
<TOPINST> Type in <HP>patron's name</HP> and <HP>card number</HP>
          (if applicable)
<TOPINST> Then select an action bar choice.
<AREA>
```

```
    <DTACOL PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25 SELWIDTH=25>
    <DTAFLD DATAVAR=curdate USAGE=out ENTWIDTH=8>Date
    <DTAFLD DATAVAR=cardno ENTWIDTH=7>Card No.
      <DTAFLDD>(A 7-digit number)
    <DTAFLD DATAVAR=name>Name
      <DTAFLDD>(Last, First, M.I.)
    <DTAFLD DATAVAR=address>Address
     </DTACOL>
  <DIVIDER>
  <REGION DIR=horiz>
  <SELFLD NAME=cardsel PMTWIDTH=30 SELWIDTH=38>Choose
  one of the following
    <CHOICE CHECKVAR=card MATCH=new>New
    <CHOICE CHECKVAR=card MATCH=renew>Renewal
    <CHOICE CHECKVAR=card MATCH=replace>Replacement
  </SELFLD>
  <SELFLD TYPE=multi PMTWIDTH=30 SELWIDTH=25>Check valid branches
    <CHOICE NAME=north HELP=nthhlp CHECKVAR=nth>North Branch
    <CHOICE NAME=south HELP=sthhlp CHECKVAR=sth>South Branch
    <CHOICE NAME=east HELP=esthlp CHECKVAR=est>East Branch
    <CHOICE NAME=west HELP=wsthlp CHECKVAR=wst>West Branch
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

```
  File  Search  Help
------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number (if applicable).

Then select an action bar choice.

Date . . . :
Card No. . . _______  (A 7-digit number)
Name . . . . _____________________________  (Last, First, M.I.)
Address  . . _____________________________

Choose one of the following          Check valid branches
__  1.  New                          _  North Branch
    2.  Renewal                      _  South Branch
    3.  Replacement                  _  East Branch
                                     _  West Branch

Enter a command ===> ____________________________________________________________
 F1=Help        F2=Split       F3=Exit        F6=KEYSHELP    F9=Swap
F12=Cancel
```

Figure 130. HP (Highlighted Phrase)

INFO (Information Region)

The INFO tag defines an information region for a panel.

WIDTH=format-width | *
This attribute determines the width the conversion utility uses to format the text in the ISPF)BODY section of the panel. If WIDTH is not the value is set to the remaining available panel (or region) width. If specified, the WIDTH value cannot be larger than the defined width of the panel (or region) minus 2 characters. For example, a WIDTH value of 58 is acceptable for an information region within a panel with a defined width of 60.

Note: You should code the WIDTH attribute if the information region is part of an application panel definition that uses horizontal region capability. The actual width used in a horizontal region is 2 characters longer than the WIDTH attribute value to provide for attribute bytes that delimit the region.

INDENT=n
This attribute defines the number of columns to indent the current information region from the current left boundary.

Description

The INFO tag defines an information region for a panel. The information region is used to display text such as paragraphs, lists, notes, examples, and figures. A typical use of the INFO tag is for the definition of text within help panels.

Conditions

- The INFO tag requires an end tag.
- You must code the INFO tag within an AREA, HELP, or PANEL definition. See "AREA (Area)" on page 223, "HELP (Help Panel)" on page 340, and "PANEL (Panel)" on page 415 for descriptions of these tags.

Nested Tags

You can code the following tags within an INFO definition:

Tag	Name	Usage	Page	Required
DIVIDER	Divider	Multiple	296	No
DL	Definition list	Multiple	298	No
FIG	Figure	Multiple	329	No
Hn	Heading	Multiple	352	No
LINES	Lines	Multiple	366	No
NOTE	Note	Multiple	400	No
NOTEL	Note List	Multiple	402	No
NT	Note	Multiple	404	No
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
SL	Simple list	Multiple	470	No
SOURCE	Source	Multiple	472	No
UL	Unordered list	Multiple	479	No
XMP	Example	Multiple	497	No

Example

The following help panel markup contains an information region. The text of the information region is defined using two P (paragraph) tags and an unordered list (UL) tag with three LI (list item) tags. Figure 131 shows the formatted result.

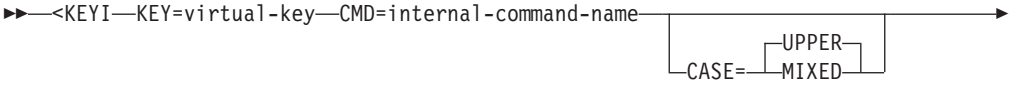
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=info WIDTH=60 DEPTH=22>ShelfBrowse Help
<AREA>
<INFO WIDTH=42>
  <P>When ShelfBrowse finds your book, it displays this
  information:
  <UL>
    <LI>Reference information about the book.
    <LI>The location of the book.
    <LI>If the book is in stock.
      <P>If the book is not in stock, see the librarian.
  </UL>
</INFO>
</AREA>
</HELP>
```

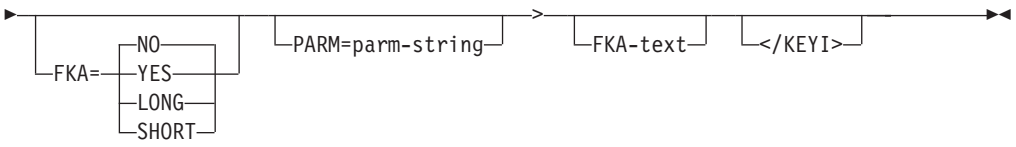
```
                    ShelfBrowse Help

When ShelfBrowse finds your book, it
displays this information:

o   Reference information about the book.

o   The location of the book.

o   If the book is in stock.

    If the book is not in stock, see the
    librarian.









 F1=Help        F3=Exit        F5=Exhelp      F6=Keyshelp
 F7=PrvTopic    F8=NxtTopic   F10=PrvPage    F11=NxtPage
F12=Cancel
```

Figure 131. Information Region

KEYI (Key Item)

The KEYI tag defines a key assignment within a key mapping list.





KEY=virtual-key
This attribute specifies the name of the key to assign to the command. The conversion utility supports F1–F24 only.

CMD=internal-command-name
This attribute specifies the command that ISPF runs when the user presses the key.

The *internal-command-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

As an extension to the Dialog Tag Language, the conversion utility supports special ISPF command syntax for *internal-command-name*. In this case, the *internal-command-name* must have the following characteristics:
- 2–9 single-byte characters in length
- The first character must be a '>', ':', or '%'.

  To code the > character you must use the ***&gtsym*** predefined entity. See "Predefined Entities" on page 25 for more information.
- The second character must be A–Z, a–z, @, #, or $.
- Remaining characters, if any, must be A–Z, a–z, @, #, $, or 0–9.

Lowercase characters are translated to their uppercase equivalents by default.

CASE=UPPER | MIXED
This attribute specifies that the *internal-command-name* will be converted to uppercase characters or stored as entered in the tag definition.

FKA=NO | YES | LONG | SHORT
This attribute specifies whether the key assignment is to appear in the function key area of an application panel. The default value NO indicates that the key is not to appear. You must specify FKA=YES, FKA=LONG, or FKA=SHORT if you want the key to be displayed in the function key area.

When FKA=NO is specified, the key is active even if it is not displayed.

Compatibility Considerations

In ISPF Version 3.1, the conversion utility supported the values NO, SHORT, and LONG for the FKA attribute. The FKA attribute values have been changed to YES and NO. FKA=YES is converted to FKA=SHORT. If you want keys to appear only when the user chooses LONG form, you must code FKA=LONG.

For compatibility between releases, the conversion utility allows SHORT or LONG.

PARM=parm-string
This attribute allows a parameter to be added to the command specified by the CMD attribute. The combined length of the command and the parameter is limited to 40 bytes. When the combined length exceeds 40 bytes, truncation of

the PARM occurs at the end of the last complete word in the *parm-string*, for a *parm-string* containing multiple words. A *parm-string* which is a single word is truncated at position 40.

FKA-text
This is the text for the key which is to appear in the function key area of the panels that refer to the key list. This text is appended to the string "Fn=" (with no intervening space) to create the displayed format. You should use initial caps for the *FKA-text* value.

If not specified, the *FKA-text* defaults to the *internal-command-name* specified for the key.

The function key area is formatted at run time based on the panel size. The maximum number of bytes allowed for *FKA-text* is 64. If the text exceeds 64 bytes, it is truncated and a warning message is issued. The conversion utility removes excess blanks from *FKA-text*. The first 8 bytes of the resulting text are used by ISPF.

Description

The KEYI tag defines a key assignment within a key mapping list. Key assignments provide a means of associating commands with keys.

KEYI tags with the same assignment cause the conversion utility to issue a warning message and retain only the first occurrence.

Conditions

- You must code the KEYI tag within a KEYL definition. See "KEYL (Key List)" on page 361 for a complete description of this tag.
- Each KEYI definition can only have one command assigned to it. Additionally, CUA requires the following conventions when assigning commands to certain keys:
  - If KEY=F1 or F13, then CMD must be HELP.
  - If KEY=F3 or F15, then CMD must be EXIT.
  - If KEY=F12 or F24, then CMD must be CANCEL.

  ISPF will allow you to provide the name of your own command on these keys.

  If you code the command HELP, EXIT, or CANCEL as part of your KEYI definition, then HELP must be assigned to key F1 or F13, EXIT must be assigned to F3 or F15, and CANCEL must be assigned to F12 or F24.

Nested Tags

None.

Example

The following source file markup contains a key mapping list and an application panel that refers to the key mapping list. The F7 and F8 keys do not appear on the panel because they both have an FKA value of NO. Figure 132 on page 361 shows the formatted application panel with the displayed keys.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampabc system>
  <!entity sampbody system>)>
&sampvar1;
```

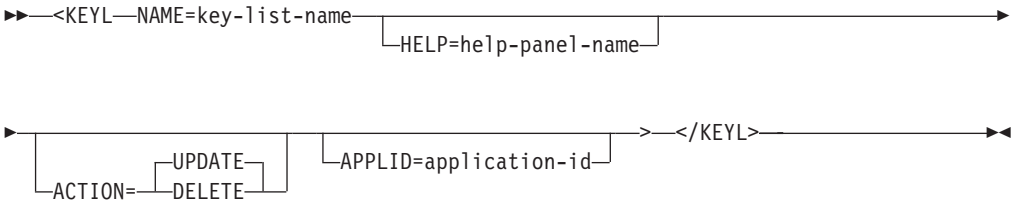
```
<KEYL NAME=keylxmp>
  <KEYI KEY=f1  CMD=help     FKA=yes>Help
  <KEYI KEY=f2  CMD=split    FKA=yes>Split
  <KEYI KEY=f3  CMD=exit     FKA=yes>Exit
  <KEYI KEY=f5  CMD=search   FKA=no>Display
  <KEYI KEY=f6  CMD=keyhlp   FKA=yes>Keyshelp
  <KEYI KEY=f7  CMD=backward FKA=no>Backward
  <KEYI KEY=f8  CMD=forward  FKA=no>Forward
  <KEYI KEY=f9  CMD=swap     FKA=yes>Swap
  <KEYI KEY=f10 CMD=actions  FKA=no>Actions
  <KEYI KEY=f12 CMD=cancel   FKA=yes>Cancel
</KEYL>

<PANEL NAME=keyi KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
&sampbody;
</PANEL>
```

```
  File  Search  Help
-------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number if applicable.

Then select an action bar choice.

Date . . . :
Card No. . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following          Check valid branches
__  1.  New                         _  North Branch
    2.  Renewal                     _  South Branch
    3.  Replacement                 _  East Branch
                                    _  West Branch

Enter a command ===> _____________________________________________________
 F1=Help       F2=Split       F3=Exit        F6=KEYSHELP    F9=Swap
F12=Cancel
```

Figure 132. Key Items

KEYL (Key List)

The KEYL tag defines a key mapping list where keys can be mapped to commands.



NAME=key-list-name
This attribute specifies a name for a key list. The HELP, HELPDEF, PANEL, and PANDEF tag refer to the *key-list-name*.

The *key-list-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

The name of the keylist table is xxxxKEYS where xxxx represents the application identifier provided to ISPDTLC with the KEYLAPPL keyword when invoked, in the "Keylist Application ID" field on the invocation panel, or with the APPLID attribute of this tag.

The *key-list-name* is used to identify the entry in the keylist table. For example, if NAME=CONVLIST and KEYLAPPL=XYZ, then CONVLIST will be written as a table entry to member XYZKEYS in the table library partitioned data set.

Keylists are updated using ISPF table services. Input is obtained from the ISPTLIB DDname allocation and output is written to the ISPTABL DDname allocation. Refer to the description of how to allocate libraries before starting ISPF in the *ISPF User's Guide* for more information about the use of ISPTLIB and ISPTABL.

See "Chapter 10. Using the Conversion Utility" on page 175 for more information on invocation parameters for the conversion utility.

HELP=help-panel-name
This attribute names a help panel that displays when the user requests help on a keylist display.

If a user requests help for a keylist and no help has been defined by the KEYL tag, the ZKEYHELP variable is checked for a help panel name. If the application has not set ZKEYHELP, a message that keyshelp is not available is displayed.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels.

ACTION=UPDATE | DELETE
This attribute specifies the type of action requested for the keylist specified by *key-list-name*.

When ACTION=DELETE is specified, it is not necessary to nest any KEYI tags within the KEYL tag definition.

APPLID=application-id
This attribute provides the application ID used to build the keylist name. The *application-id* overrides the KEYLAPPL invocation option value.

Description

The KEYL tag defines a key mapping list where keys can be mapped to commands.

To display these keys on a panel requires that the PANEL or PANDEF tag refer to the *key-list-name*. ISPF uses the specified key mapping list when building the display dependent on the user's setting by the FKA command. For more information on displaying and formatting of the function key area, refer to the appropriate section in the *ISPF User's Guide*

Conditions

- The KEYL tag requires an end tag.
- The KEYL tag cannot be nested within any other tag definition.
- When ACTION=UPDATE is specified (or defaulted), at least one KEYI tag must be included in the keylist definition.

Nested Tags

You can code the following tag within a KEYL definition:

Tag	Name	Usage	Page	Required
KEYI	Key item	Multiple	358	No

Example

The following source file markup contains a key mapping list and an application panel that refers to the key mapping list. Figure 133 on page 364 shows the formatted application panel with the displayed keys.

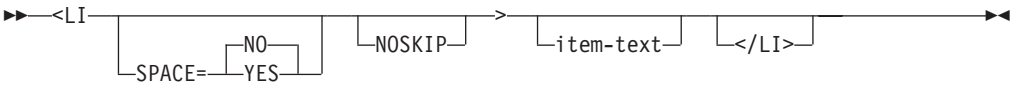
```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampabc system>
  <!entity sampbody system>)>
&sampvar1;

<KEYL NAME=keyltbl>
  <KEYI KEY=f1  CMD=help     FKA=yes>Help
  <KEYI KEY=f2  CMD=split    FKA=yes>Split
  <KEYI KEY=f3  CMD=exit     FKA=yes>Exit
  <KEYI KEY=f5  CMD=search   FKA=no>Display
  <KEYI KEY=f6  CMD=keyhlp   FKA=no>Keyshelp
  <KEYI KEY=f7  CMD=backward FKA=yes>Backward
  <KEYI KEY=f8  CMD=forward  FKA=yes>Forward
  <KEYI KEY=f9  CMD=swap     FKA=yes>Swap
  <KEYI KEY=f10 CMD=actions  FKA=no>Actions
  <KEYI KEY=f12 CMD=cancel   FKA=yes>Cancel
</KEYL>

<PANEL NAME=keyl KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
&sampbody;
</PANEL>
```

```
  File  Search  Help
------------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number if applicable.

Then select an action bar choice.

Date . . . :
Card No. . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following          Check valid branches
__  1.  New                        _  North Branch
    2.  Renewal                    _  South Branch
    3.  Replacement                _  East Branch
                                   _  West Branch

Enter a command ===> ___________________________________________________________
 F1=Help       F2=Split       F3=Exit        F6=KEYSHELP    F9=Swap
F12=Cancel
```

Figure 133. Function Keys

LI (List Item)

The LI tag defines a list item within a note list, ordered list, unordered list, or simple list.



SPACE=NO | YES
The SPACE attribute controls the indentation space for the list item. When the SPACE attribute is not specified on the LI tag, the SPACE attribute from the enclosing list tag is used to set the indentation space for the *item-text*.

When SPACE=YES, the indentation is set to 3 spaces. When SPACE=NO (or SPACE is not specified), the indentation is set to 4 spaces.

The SPACE attribute can be used to control the alignment of list items when the first word of some list items is a DBCS word preceded by a shift-out character and the first word of other list items is a SBCS word.

NOSKIP
This attribute causes the list item to format without creating a blank line before the item.

item-text
This is the text of the list item.

Description

The LI tag defines a list item within a note list, ordered list, unordered list, or simple list.

The formatting of the LI tag is dependent on the type of list you use it within and the level of nesting.

List	Formatting
Note	Formats with a 3-space or 4-space indentation (depending on the SPACE attribute) and is preceded by sequential numbers.
Ordered	Formats with a 3-space or 4-space indentation (depending on the SPACE attribute) within the level of the list in which it is defined and is preceded by sequential numbers or letters.
Simple	Formats with a 3-space or 4-space indentation (depending on the SPACE attribute) within the level of the list it is defined within.
Unordered	Formats with a 3-space or 4-space indentation (depending on the SPACE attribute) within the level of the list in which it is defined and is preceded by bullets or dashes.

The next list item implicitly ends the previous list item as do the NOTEL, OL, SL, and UL end tags.

If you do not specify text for a list item, a blank line is displayed for that item.

Conditions

- You must code the LI tag within a NOTEL, OL, SL, or UL definition. See "NOTEL (Note List)" on page 402, "OL (Ordered List)" on page 405, "SL (Simple List)" on page 470, and "UL (Unordered List)" on page 479 for descriptions of these tags.

Nested Tags

You can code the following tags within an LI definition:

Tag	Name	Usage	Page	Required
ATTENTION	Attention	Single	232	No
CAUTION	Caution	Single	240	No
DL	Definition list	Multiple	298	No
FIG	Figure	Multiple	329	No
HP	Highlighted phrase	Multiple	354	No
LINES	Lines	Multiple	366	No
NOTE	Note	Multiple	400	No
NOTEL	Note List	Multiple	402	No
NT	Note	Multiple	404	No
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No
WARNING	Warning	Single	491	No

Tag	Name	Usage	Page	Required
XMP	Example	Multiple	497	No

Example

The following help panel markup contains an unordered list with three list items. The last list item contains an additional paragraph of text. Figure 134 shows the formatted result.

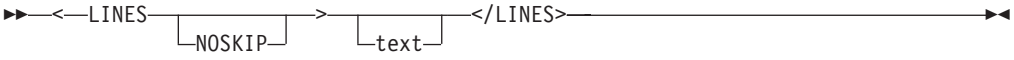
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=li DEPTH=20>ShelfBrowse Help
<AREA>
<INFO>
  <P>When ShelfBrowse finds your book,
  it displays this information:
  <UL>
    <LI>Reference information about the book.
    <LI>The location of the book.
    <LI>If the book is in stock.
      <P>If the book is not in stock, see the librarian.
  </UL>
  <P>Thank you for using ShelfBrowse.
</INFO>
</AREA>
</HELP>
```

```
                ShelfBrowse Help

When ShelfBrowse finds your book, it displays
this information:

o   Reference information about the book.

o   The location of the book.

o   If the book is in stock.

    If the book is not in stock, see the
    librarian.

Thank you for using ShelfBrowse.


 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 134. List Items

LINES (Lines)

The LINES tag defines unformatted text within an information region.



NOSKIP
This attribute causes the blank line normally placed before the lines to be skipped.

text
This is the unformatted text.

Description

The LINES tag defines unformatted text within an information region. Tags that normally cause word-wrapping (such as the P, LI, or CAUTION) do not cause wrapping when nested within a LINES definition.

If the source text on any line is too long to fit in the remaining available formatting width, the conversion utility truncates that line. The conversion utility issues a warning message the first time that truncation occurs.

The formatting of the LINES tag is similar to that of the FIG tag, except that there is no border or caption capability.

Conditions

- The LINES tag requires an end tag.
- You must code the LINES tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.

Nested Tags

You can code the following tags within a LINES definition:

Tag	Name	Usage	Page	Required
DL	Definition list	Multiple	298	No
HP	Highlighted phrase	Multiple	354	No
NOTE	Note	Multiple	400	No
NOTEL	Note List	Multiple	402	No
NT	Note	Multiple	404	No
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No
XMP	Example	Multiple	497	No

Example

The following application panel markup contains a LINES definition. The formatted output of the LINES definition is identical to the input markup. Figure 135 on page 368 shows the formatted results.

```
<!DOCTYPE DM SYSTEM>

<PANEL NAME=lines DEPTH=22 WIDTH=40>Lines Tag Example
```

```
<AREA>
<INFO WIDTH=38>
<P>The following area shows how the LINES tag formats.
<LINES>
First line, just at it was entered.
  Second line, ditto.

Notice we skipped a line here?

  You
     can
        even
            do
              this.
</LINES>
<P>The LINES tag formatting ends immediately above.
</INFO>
</AREA>
</PANEL>
```

```
          Lines Tag Example

The following area shows how the LINES
tag formats.

First line, just at it was entered.
  Second line, ditto.

Notice we skipped a line here?

  You
     can
        even
            do
              this.

The LINES tag formatting ends
immediately above.



 F1=Help    F3=Exit   F12=Cancel
```

Figure 135. LINES

LIT (Literal)

The LIT tag defines a string where all blanks are significant and included in the value.

literal-display-value
This attribute specifies a string with all blanks preserved.

Description

The LIT tag defines a string where all blanks are significant and included in the value. No stripping of leading, trailing, or embedded blanks is performed.

This is the only way to specify trailing blanks or a value of all blanks in the XLATI *displayed-value*.

The LIT start and end tags must be on the same line as the *literal-display-value* to preserve the original spacing of the value.

Conditions

- The LIT tag requires an end tag.
- You must code the LIT tag only within an XLATI definition. See "XLATI (Translate Item)" on page 493 for a complete description of this tag.
- Multiple LIT tags may be coded within a single XLATI definition, as long as they are not nested within each other. However, a better approach is to include the whole XLATI *displayed-value* within the LIT tag.

Nested Tags

None.

Example

The following markup contains a variable class definition with two translate lists. The last four translate items in the second list contain LIT definitions that preserve trailing blanks in the displayed value of their respective translate items.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=aa TYPE='char 2'>
<VARCLASS NAME=bb TYPE='char 9'>
<VARCLASS NAME=cc TYPE='char 9'>
  <XLATL FORMAT=upper>
  </XLATL>
  <XLATL>
    <XLATI VALUE=1>BIGCHARGE
    <XLATI VALUE=2><LIT>V I S T A</LIT>
    <XLATI VALUE=3><LIT>EZCARD   </LIT>
    <XLATI VALUE=4><LIT>CHECK    </LIT>
    <XLATI VALUE=5><LIT>     CASH</LIT>
</XLATL>

<VARLIST>
  <VARDCL NAME=dispva VARCLASS=cc>
  <VARDCL NAME=inptva VARCLASS=bb>
  <VARDCL NAME=chckva VARCLASS=aa>
</VARLIST>

<PANEL NAME=lit>LIT translation
  <TOPINST>You can display this panel with ISPF option 7.2
  <TOPINST>For this example, enter the word 'BIGCHARGE', 'V I S T A',
           'EZCARD', 'CHECK', or '     CASH' in the "literal value"
           field (no quotes).
  <TOPINST>The literal will be translated to the corresponding number
           defined in the XLATL tag, and will be displayed in the
           "translated value" field.
  <TOPINST>The literal you enter will be displayed (left justified) in
           the "original value" field.
  <DTACOL>
  <:-- assign "literal value" to "original value" -->
  <SOURCE>
```

```
  &inptva = &dispva
  </SOURCE>
  <DTAFLD DATAVAR=dispva ENTWIDTH=9 PMTWIDTH=20 ALIGN=center>Literal value
  <DTAFLD DATAVAR=chckva ENTWIDTH=2 PMTWIDTH=20 USAGE=out>Translated value
  <DTAFLD DATAVAR=inptva ENTWIDTH=9 PMTWIDTH=20 USAGE=out>Original value
  <:-- assign translated "literal value" to "translated value" -->
  <SOURCE>
  &chckva = &dispva
  </SOURCE>
  </DTACOL>
  <CMDAREA>
</PANEL>
```

LP (List Part)

The LP tag defines a comment or explanation within a note list, ordered list, unordered list, or simple list.

NOSKIP
This attribute causes the list part to format without creating a line before the list part.

implied-paragraph
This is the text of the list part.

Description

The LP tag defines a comment or explanation within an ordered list, unordered list, or simple list. You can code the LP tag anywhere within a list.

The text of the list part starts at the left margin of the current level of the list. It is not numbered or lettered. When you use it within a NOTEL or OL definition, LP does not interrupt or increment the sequence.

The next list item or the end of the list implicitly ends the list part.

Conditions

- You must code the LP tag within a NOTEL, OL, SL, or UL definition. See "NOTEL (Note List)" on page 402, "OL (Ordered List)" on page 405, "SL (Simple List)" on page 470, and "UL (Unordered List)" on page 479 for descriptions of these tags.

Nested Tags

You can code the following tags within an LP definition:

Tag	Name	Usage	Page	Required
ATTENTION	Attention	Single	232	No
CAUTION	Caution	Single	240	No
DL	Definition list	Multiple	298	No
FIG	Figure	Multiple	329	No
HP	Highlighted phrase	Multiple	354	No

Tag	Name	Usage	Page	Required
LINES	Lines	Multiple	366	No
NOTE	Note	Multiple	400	No
NOTEL	Note List	Multiple	402	No
NT	Note	Multiple	404	No
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No
WARNING	Warning	Single	491	No
XMP	Example	Multiple	497	No

Example

The following help panel markup contains an ordered list with a nested list part tag. WARNING and P tags are nested within the list part definition. Figure 136 on page 372 shows the formatted result.

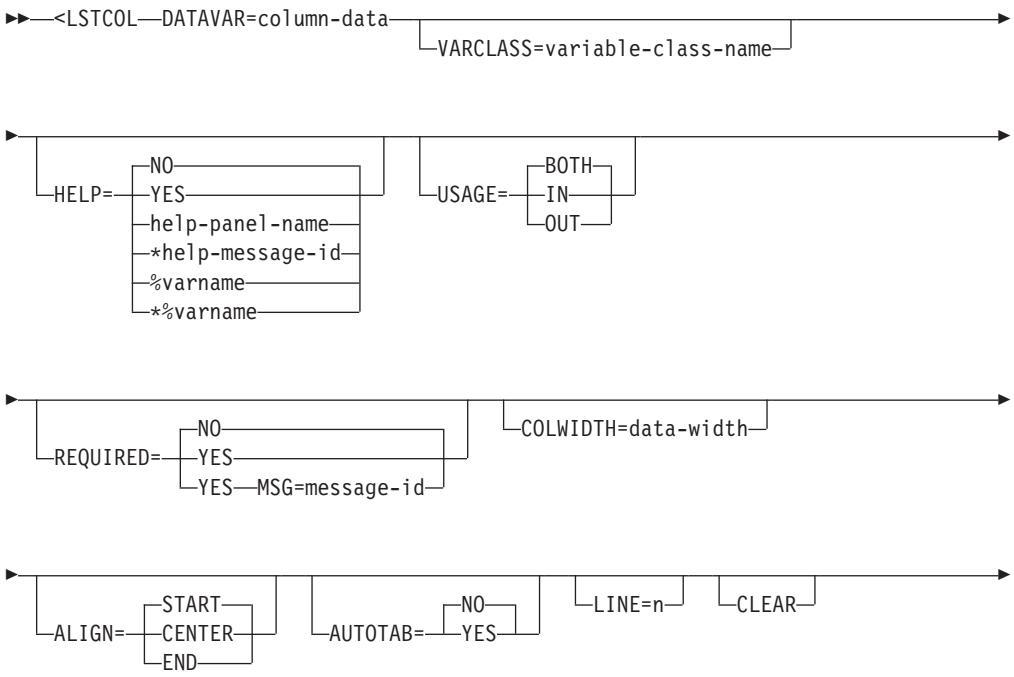
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=lp WIDTH=50 DEPTH=20>Help For Changing a File
<AREA>
<INFO>
  <OL>
    <LI>Type over the existing data
    in the entry fields with the new data.
    <LP>
      <WARNING>
      Performing the next step will save all changes
      and delete the existing data.
      <P>To quit this function without
      deleting the existing data, press F12=Cancel.
      </WARNING>
    <LI>Press Enter to save the
    updated data.
  </OL>
</INFO>
</AREA>
</HELP>
```

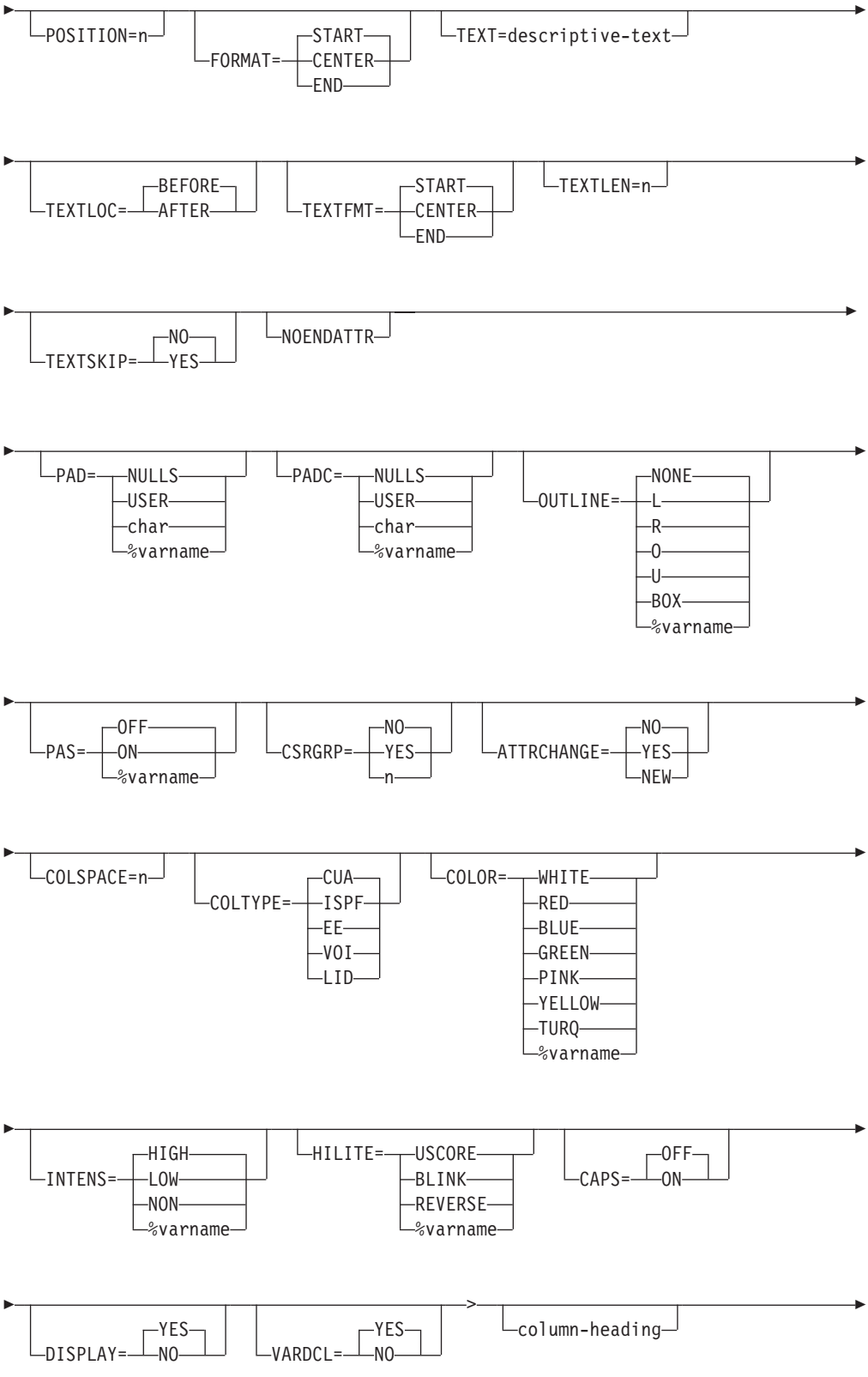
```
            Help For Changing a File

1.  Type over the existing data in the entry
    fields with the new data.

Warning: Performing the next step will save all
changes and delete the existing data.

To quit this function without deleting the
existing data, press F12=Cancel.

2.  Press Enter to save the updated data.




 F1=Help          F3=Exit          F5=Exhelp
 F6=Keyshelp      F7=PrvTopic      F8=NxtTopic
F10=PrvPage      F11=NxtPage      F12=Cancel
```

Figure 136. List Part

LSTCOL (List Column)

The LSTCOL tag defines a column of data from an ISPF table displayed in the ISPF table display area of a panel.




POSITION=n
FORMAT=
START
CENTER
END
TEXT=descriptive-text
TEXTLOC=
BEFORE
AFTER
TEXTFMT=
START
CENTER
END
TEXTLEN=n
TEXTSKIP=
NO
YES
NOENDATTR
PAD=
NULLS
USER
char
%varname
PADC=
NULLS
USER
char
%varname
OUTLINE=
NONE
L
R
O
U
BOX
%varname
PAS=
OFF
ON
%varname
CSRGRP=
NO
YES
n
ATTRCHANGE=
NO
YES
NEW
COLSPACE=n
COLTYPE=
CUA
ISPF
EE
VOI
LID
COLOR=
WHITE
RED
BLUE
GREEN
PINK
YELLOW
TURQ
%varname
INTENS=
HIGH
LOW
NON
%varname
HILITE=
USCORE
BLINK
REVERSE
%varname
CAPS=
OFF
ON
DISPLAY=
YES
NO
VARDCL=
YES
NO
>
column-heading

DATAVAR=column-data
This is the data which will occupy the column. The *column-data* value must be an ISPF table variable name (without a leading % sign).

VARCLASS=variable-class-name
This is the name of a variable class, defined with a VARCLASS tag, that overrides the default variable class referred to by the VARDCL tag that declares the data variable for the list column.

HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname
This attribute specifies the help action taken when the user requests help for this list column. This is field-level help.

When HELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help for a list column and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

USAGE=BOTH | IN | OUT
This attribute indicates if this column is for input, output, or both.

REQUIRED=NO | YES
This attribute indicates if this column is required to have input for each modified row. The default is NO. This attribute is valid only when USAGE=IN or BOTH.

If you specify REQUIRED=YES, a conditional VER(variable,NONBLANK) statement will be built by the conversion utility and placed in the)PROC section of the ISPF panel generated. This results in the column variable being verified only when the row is selected or modified.

MSG=message-id
This attribute specifies the message that is displayed when the user does not complete a required entry (defined with the REQUIRED attribute). If you do not specify a *message-identifier*, ISPF displays a default message.

If you specify the MSG attribute and REQUIRED=YES, a VER(variable,NONBLANK,MSG=message-identifier) statement is built by the conversion utility and placed in the)PROC section of the ISPF panel generated. If you specify the MSG attribute and REQUIRED=NO (the default), the conversion utility issues a warning message.

See "MSG (Message)" on page 393 for information on creating messages.

Note: You can specify messages pertaining to other validations using XLATL and CHECKL tags within a VARCLASS definition. See the descriptions of these tags for additional information.

COLWIDTH=data-width
This attribute determines the data width to be used by the column. If you do not specify this attribute, the data width and column formatting width are determined by the actual length of the *column-heading* If the width of the *column-heading* text is greater than the COLWIDTH, it will be used as the column formatting width. The minimum width is 1 and the maximum is the remaining available panel (or region) width. If the *column-heading* and the COLWIDTH attribute are omitted, the data width and column formatting width are determined by the TYPE value of the associated VARCLASS. If a VARCLASS TYPE value is not available, the size of the column variable name (specified by the DATAVAR attribute) determines the width.

You should code the COLWIDTH attribute with a value equal to the length of the table data variable.

ALIGN=START | CENTER | END
This attribute specifies how the data value is to be displayed in the data field.

An attribute character is used for the field that specifies JUST(LEFT) if ALIGN=START, JUST(ASIS) if ALIGN=CENTER or JUST(RIGHT) if ALIGN=END. When ALIGN=END, there is no underscore padding performed; blanks are used.

AUTOTAB=NO | YES
When AUTOTAB=YES, the cursor moves to the next field capable of input when the user enters the last character in the list column field.

AUTOTAB=YES is valid only when the value for USAGE is either BOTH or IN.

The ISPF SKIP keyword is not supported when running in GUI mode.

LINE=n
This attribute is an provides the application the ability to place LSTCOL fields on different model lines. ISPF defines the range of lines as 1 to 8. The default is 1. Column headings are generated on multiple lines to match the LSTCOL field placement.

CLEAR
This attribute adds a CLEAR (variable, ...) statement to the)MODEL line. CLEAR should be specified for table extension variables.

For more information about the)MODEL line, refer to the *ISPF User's Guide*

POSITION=n
This attribute specifies the starting position of the data column and related text or the column heading, if the heading is longer than the data column. If this attribute is not specified or is not valid, the conversion utility formats the column immediately to the right of the previous column on the specified or default model line. This attribute allows you to position fields on different model lines with vertical alignment. Column position is location of the attribute byte preceding the data column.

FORMAT=START | CENTER | END
This attribute specifies how the data column and its column heading are formatted. If you do not specify this attribute, or if you specify the attribute value START, then the column formats as in ISPF Version 3.1 and ISPF Version 3.2.

Formatting of the data in the column takes place within the column width, which is determined as described in the COLWIDTH attribute section.

When you specify the attribute value CENTER, the conversion utility centers a column heading that is shorter than the column width. If the column heading is longer than the column width, then the data column is centered under the column heading. When either the heading or the data column is centered, blank characters are added before and after the column heading or data column. The total amount of space to be added is divided by 2 and the resulting whole number is the number of blanks added in front of the column heading or data column. The difference between the total amount of space and the amount placed in front of the column heading or data column is used at the end.

When you specify the attribute value END, a column heading that is shorter than the column width is right-justified so it aligns with the end of the displayed data. If the column heading is longer than the column width, the data column is right-justified so that the displayed data and the column heading end at the same position.

If there is insufficient space available to format the column heading as requested, the conversion utility issues a message that the FORMAT attribute is ignored.

The FORMAT attribute does not affect the display of the field contents within the data column, which is determined by the ALIGN attribute.

TEXT=descriptive-text
This attribute specifies a short description of the data column. It can be placed before or after the data column. Text containing special characters or imbedded blanks must be enclosed in quotes.

TEXTLOC=BEFORE | AFTER
This attribute specifies the location of the TEXT relative to the data column. Text can be placed on either side of the data column.

TEXTFMT=START | CENTER | END
This attribute specifies the format of the text within the length of the text area. The text can be left-justified, centered, right-justified.

TEXTLEN=n
This attribute specifies the amount of space to reserve for formatting the descriptive text. This attribute helps you line up text on different model lines, and if the space reserved is longer than the descriptive text, it permits formatting within the reserved space with the TEXTFMT attribute. If the descriptive text is longer than the space reserved by the TEXTLEN attribute, the descriptive text is not formatted and a warning message is issued.

TEXTSKIP=NO | YES
This attribute provides for skipping past the *descriptive text* when either the TEXTLOC=BEFORE and the previous LSTCOL tag includes the NOENDATTR attribute, or TEXTLOC=AFTER and the current LSTCOL tag includes the NOENDATTR attribute. If there is no other input field on the panel, the cursor moves to the first input field. The ISPF SKIP keyword is not supported in GUI mode.

NOENDATTR
This attribute specifies that no ending attribute character will be placed after the data column. NOENDATTR is ignored for the last data column on each model line.

PAD=NULLS | USER | char | %varname
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

PADC=NULLS | USER | char | %varname
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

OUTLINE=NONE | L | R | O | U | BOX | %varname
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

PAS=OFF | ON | %varname
This attribute controls the availability of the point-and-shoot function for this table field. You can define this attribute as a variable name preceded by "%"

CSRGRP=NO | YES | n
When CSRGRP=YES, the conversion utility generates a cursor group number to be used for this table column. When CSRGRP=n, the number provided is used for this field. The PAS attribute must be specified as ON or %varname.

The conversion utility accepts the CSRGRP attribute for any table field definition. The CRSGRP attribute is used at run-time for output fields only.

ATTRCHANGE=NO | YES | NEW
When ATTRCHANGE=YES or ATTRCHANGE=NEW, the conversion utility formats an additional entry in the panel)ATTR section (that can apply to multiple list columns) instead of creating a unique ".ATTR(field-name)" entry in the)INIT section for this field. With this option, multiple LSTCOL tags with the same characteristics require fewer panel logic statements. ATTRCHANGE=NEW creates a new entry. ATTRCHANGE=YES uses an existing entry, if possible.

COLSPACE=n
The COLSPACE attribute specifies the total number of bytes for the column width, including the leading and trailing attributes, and any trailing blank following an input field. The use of the COLSPACE attribute causes column heading text longer than the COLSPACE value (minus the attribute bytes) to be flowed into multiple lines.

COLTYPE=CUA | ISPF | EE | VOI | LID
This attribute defines the attribute type to be applied to the table field. TYPE=CUA, the default, causes the field to display using the standard CUA attribute.

VOI and LID are valid only when USAGE=OUT.

EE is valid when USAGE=IN or USAGE=BOTH.

Using a CUA attribute causes the field to appear in the associated. color.

When COLTYPE=ISPF, a non-CUA attribute is generated and you can specify the color, intensity, and highlighting of the field using the COLOR, INTENS, and HILITE attributes. These attributes are not valid for CUA types.

COLOR=WHITE | RED | BLUE | GREEN | PINK | YELLOW | TURQ | %varname
This attribute specifies the color of the field. You can define this attribute as a variable name preceded by a percent (%) sign.

INTENS=HIGH | LOW | NON | %varname
This attribute defines the intensity of the field. You can define this attribute as a variable name preceded by a percent (%) sign.

HILITE=USCORE | BLINK | REVERSE | %varname
This attribute specifies the extended highlighting attribute of the field. You can define this attribute as a variable name preceded by a percent (%) sign.

CAPS=OFF | ON
When CAPS=ON, the data in the field is displayed in uppercase characters.

DISPLAY=YES | NO
This attribute specifies whether the data for the field will be visible when the panel is displayed. This attribute is used to allow fields to contain information you do not want to appear on the screen.

VARDCL=YES | NO
When VARDCL=NO the list column name is not checked to the declared variable information provided with the VARCLASS and VARDCL tags.

column-heading
This is the text of the list column heading. If the length of the *column-heading* and the COLWIDTH values are not equal, the greater of the two is used to determine column formatting width. If the column-heading and the COLWIDTH attributes are omitted, the column formatting width is determined by the TYPE value of the associated VARCLASS. If a VARCLASS TYPE value is not available, the size of the column variable name (specified by the DATAVAR attribute) determines the width.

The *column-heading* text placement over the column is determined by the FORMAT attribute value.

Description

In conjunction with the LSTFLD tag, LSTCOL tags provide a means of defining a vertically scrollable list display area that is made up of columns of data coming from ISPF table data. One or more ISPF)MODEL section statements will be built to display the fields defined by the LSTCOL tags. The use of LSTCOL tags requires the use of the TBDISPL service in the application program.

If the ISPF panel width is smaller than the total width of the group of columns, columns that exceed the panel width will be clipped from the right. A warning message is issued if this condition occurs.

You can use the LINE attribute to format your table to display on multiple lines.

If NOENDATTR is not specified, the conversion utility generates a beginning and ending attribute for each column of the table display)MODEL line. An additional blank is also inserted for fields with USAGE=IN or BOTH if AUTOTAB=NO. This characteristic results in the following conditions:

- When USAGE=OUT, 2 extra spaces are added to the defined column formatting width.
- When AUTOTAB=YES and USAGE=IN or BOTH, 2 extra spaces are added to the defined column formatting width.
- When AUTOTAB=NO (the default) and USAGE=IN or BOTH, 3 extra spaces are added to the defined column formatting width.

It is important that you allow for this extra space when designing your panel. The extra space is added to the width value for the field as defined in the description of the COLWIDTH attribute.

Conditions

- You must code the LSTCOL tag within a LSTFLD or LSTGRP tag. See "LSTFLD (List Field)" on page 382 for a complete description of this tag.
- Each LSTCOL definition should have a VARDCL definition associated with the variable value specified with the DATAVAR attribute. See "VARDCL (Variable Declaration)" on page 486 for a complete description of this tag.
- Only MODEL lines that actually are formatted with fields are written in the panel body. Thus, if some LSTCOL entries specify LINE=1 and others specify LINE=3, but there are no LSTCOL entries for LINE 2, only two MODEL lines are created.
- If both PAD and PADC have been specified, PAD is ignored and PADC is used.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested Tags

You can code the following tags within a LSTCOL definition:

Tag	Name	Usage	Page	Required
COMMENT	Comment	Multiple	282	No
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SOURCE	Source	Multiple	472	No

Examples

In the following source file markup, the application panel contains a list field with five list columns. The first three columns are defined as output-only, and are coded within the **Subscriber Name** list group. The **Number** column is an input/output column, and it is coded within the **Phone** list group. The last column is input-only, and it is coded within the **Approved** list group. This column requires input, so if it is not filled in, the error message "MSGG886" will be displayed. The variable declarations and classes associated with the list columns are also shown. Figure 137 on page 380 shows the formatted result of the application panel.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=namecls TYPE='char 15'>
<VARCLASS NAME=midcls  TYPE='char 1'>
<VARCLASS NAME=phoncls TYPE='char 12'>
<VARCLASS NAME=appcls  TYPE='char 1'>
  <XLATL FORMAT=upper>
  </XLATL>
  <CHECKL>
    <CHECKI TYPE=values PARM1=EQ PARM2='Y N'>
  </CHECKL>

<VARLIST>
  <VARDCL NAME=xfname  VARCLASS=namecls>
  <VARDCL NAME=xlname  VARCLASS=namecls>
```

```
  <VARDCL NAME=xmid    VARCLASS=midcls>
  <VARDCL NAME=xphone  VARCLASS=phoncls>
  <VARDCL NAME=xapp    VARCLASS=appcls>
</VARLIST>

<PANEL NAME=lstcola KEYLIST=keyltbl>Subscriber List
<TOPINST>Enter phone number, if missing,
(format - nnn-nnn-nnnn) and approved
indicator (y or n) for each person.
<AREA>
  <LSTFLD>
    <LSTGRP HEADLINE=yes>Subscriber Name
      <LSTCOL DATAVAR=xfname USAGE=out COLWIDTH=15>First Name
      <LSTCOL DATAVAR=xlname USAGE=out COLWIDTH=15>Last Name
      <LSTCOL DATAVAR=xmid   USAGE=out COLWIDTH=1>MI
    </LSTGRP>
    <LSTGRP>Phone
      <LSTCOL DATAVAR=xphone COLWIDTH=12>Number
    </LSTGRP>
    <LSTGRP>Approved
      <LSTCOL DATAVAR=xapp USAGE=in REQUIRED=yes
        COLWIDTH=1 MSG=msgg886>(Y or N)
    </LSTGRP>
  </LSTFLD>
</AREA>
<CMDAREA>
</PANEL>
```

```
                            Subscriber List               ROW 1 TO 3 OF 3

Enter phone number, if missing, (format - nnn-nnn-nnnn) and approved
indicator (y or n) for each person.

--------- Subscriber Name ----------  Phone          Approved

First Name       Last Name        MI  Number         (Y or N)
Pete             Moss             P   919-888-4444   _
Sally            Forth            N   ____________   _
Melba            Toast            T   919-444-8888   _
****************************** BOTTOM OF DATA ******************************









Command ===> ____________________________________________________________
 F1=Help       F2=Split      F3=Exit       F7=Backward     F8=Forward
 F9=Swap       F12=Cancel
```

Figure 137. List Columns

To display the same table in a different format, we can change the LSTCOL tags for name to include the LINE attribute. The DTL changes are reflected in the following example.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=namecls TYPE='char 15'>
<VARCLASS NAME=midcls  TYPE='char 1'>
```

```
<VARCLASS NAME=phoncls TYPE='char 12'>
<VARCLASS NAME=appcls  TYPE='char 1'>
  <XLATL FORMAT=upper>
  </XLATL>
  <CHECKL>
    <CHECKI TYPE=values PARM1=EQ PARM2='Y N'>
  </CHECKL>

<VARLIST>
  <VARDCL NAME=xfname  VARCLASS=namecls>
  <VARDCL NAME=xlname  VARCLASS=namecls>
  <VARDCL NAME=xmid    VARCLASS=midcls>
  <VARDCL NAME=xphone  VARCLASS=phoncls>
  <VARDCL NAME=xapp    VARCLASS=appcls>
</VARLIST>

<PANEL NAME=lstcolb KEYLIST=keyltbl>Subscriber List
<TOPINST>Enter phone number, if missing,
(format - nnn-nnn-nnnn) and approved
indicator (y or n) for each person.
<AREA>
  <LSTFLD DIV=solid>
    <LSTGRP HEADLINE=yes>Subscriber Name
      <LSTCOL DATAVAR=xfname USAGE=out LINE=1 COLWIDTH=15>First Name
      <LSTCOL DATAVAR=xlname USAGE=out LINE=2 COLWIDTH=15>Last Name
      <LSTCOL DATAVAR=xmid   USAGE=out LINE=3 COLWIDTH=1>MI
    </LSTGRP>
    <LSTGRP>Phone
      <LSTCOL DATAVAR=xphone COLWIDTH=12>Number
    </LSTGRP>
    <LSTGRP>Approved
      <LSTCOL DATAVAR=xapp USAGE=in REQUIRED=yes
        COLWIDTH=1 MSG=msgg886>(Y or N)
    </LSTGRP>
  </LSTFLD>
</AREA>
<CMDAREA>
</PANEL>
```

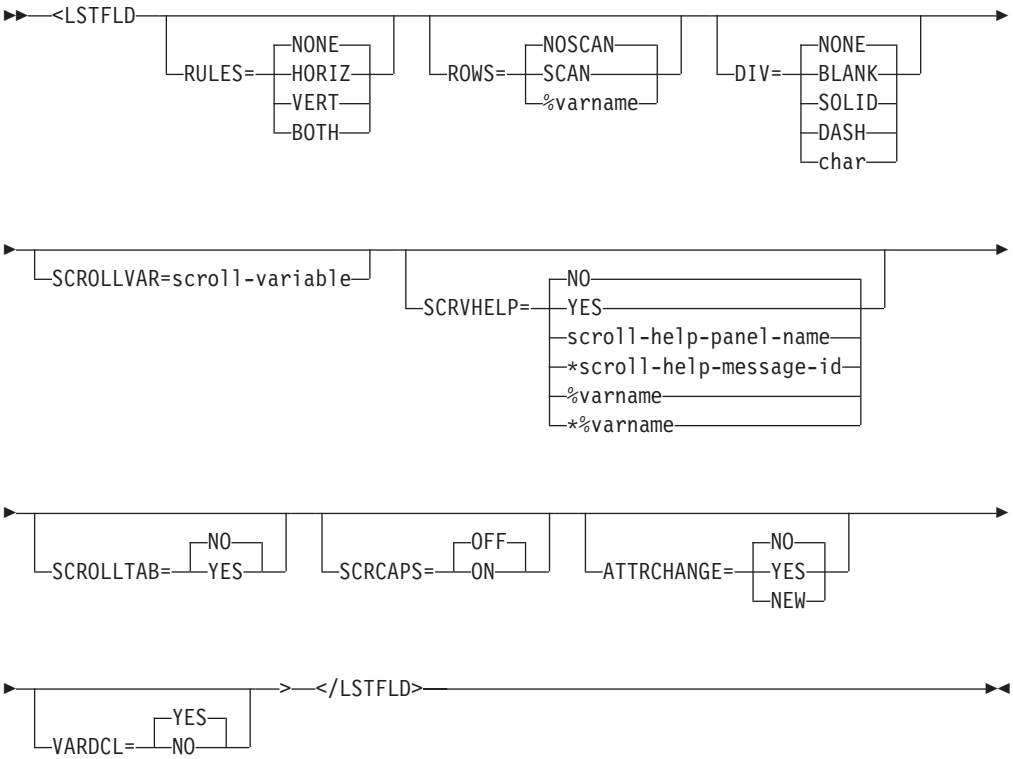
Figure 138 on page 382 shows the formatted result of the application panel.

```
                              Subscriber List                ROW 1 TO 3 OF 3

Enter phone number, if missing, (format - nnn-nnn-nnnn) and approved
indicator (y or n) for each person.

Subscriber Name  Phone          Approved
First Name       Number         (Y or N)
Last Name
MI
Pete             919-888-4444   _
Moss
P
--------------------------------------------------------------------------------
Sally            ____________   _
Forth
N
--------------------------------------------------------------------------------
Melba            919-444-8888   _
Toast
Command ===> ____________________________________________________________________
 F1=Help        F2=Split       F3=Exit        F7=Backward     F8=Forward
 F9=Swap       F12=Cancel
```

Figure 138. List Columns

LSTFLD (List Field)

The LSTFLD tag defines an ISPF table display area that is made up of columns of data coming from ISPF tables.



RULES=NONE | HORIZ | VERT | BOTH
This attribute specifies the type of interior rules that will be drawn in the table display being defined within the LSTFLD tag. This applies to all the list columns within the context of this tag.

This attribute is supported by using the ISPF outline (L | R | O | U | Box | None) statement on panel definition statements. However, the lines around fields will only be visible on double-byte character support terminals.

Note: Any list column field within the list field defining OUTLINE overrides the LSTFLD RULES value.

ROWS=NOSCAN | SCAN | %varname
This attribute provides support by TBDISPL of rows previously selected by the TBSARG service. If you specify ROWS=SCAN, the conversion utility adds ROWS(SCAN) to the)MODEL line statement in the generated ISPF panel.

If you specify ROWS=%varname, ROWS(&varname) is added to the)MODEL line. The application must set the variable name to ALL or SCAN before the panel is displayed.

DIV=NONE | BLANK | SOLID | DASH | char
This attribute specifies the type of divider line to be added as the last line of a model set. If this attribute is omitted or specified as NONE, the divider line is not generated. If this attribute is specified as BLANK, a blank divider line is generated. You may specify either SOLID or DASH to produce a visible divider line. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line. Alternately, you can specify a character or a character string of your choice. The character or characters provided are replicated to the available width of the panel (or region) to create the divider line.

If you have defined LSTCOL tags for all 8 of the available model lines, then the conversion utility issues a message and does not generate any divider line.

SCROLLVAR=scoll-variable
This attribute specifies the name of a variable that the application uses to obtain scrolling information. The *scroll-variable* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

If the attribute is specified, the conversion utility creates a scroll entry on the command line, providing that the resulting command area allows at least 8 bytes for a command entry.

SCRVHELP=NO | YES | scroll-help-panel-name | *scroll-help-message-id | %varname | *%varname
This attribute specifies the help action taken when the user requests help for the field specified with the SCROLLVAR attribute.

When SCRVHELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help on a choice and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *scroll-help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

SCROLLTAB=NO | YES
When SCROLLTAB=YES, the cursor moves to the next input field when the user enters the last character in the scroll amount field. If there is no other input field on the panel, the cursor moves to the beginning of the command line. The ISPF SKIP keyword is not supported in GUI mode.

SCRCAPS=OFF | ON
When SCRCAPS=ON, the data in the scroll field is displayed in uppercase characters.

ATTRCHANGE=NO | YES | NEW
When ATTRCHANGE=YES or ATTRCHANGE=NEW, the conversion utility formats an additional entry in the panel)ATTR section (that can apply to multiple list columns) instead of creating a unique ".ATTR(field-name)" entry in the)INIT section for each field. With this option, multiple LSTCOL tags with the same characteristics require fewer panel logic statements. ATTRCHANGE=NEW creates a new entry. ATTRCHANGE=YES uses an existing entry, if possible.

Note: Any list column field within the list field defining ATTRCHANGE overrides the LSTFLD ATTRCHANGE value.

VARDCL=YES | NO
When VARDCL=NO, the list field name is not checked to the declared variable information porvided with the VARCLASS and VARDCL tags.

Note: Any list column field within the list field defining VARDCL overrides the LSTFLD VARDCL value.

Description

The LSTFLD tag defines a scrollable list display area that is made up of columns of data coming from ISPF table data. The conversion utility creates a)MODEL line at the bottom of the)BODY section of the panel the list field is coded within.

The use of the LSTFLD tag causes all other tags that generate panel data and that are coded after the LSTFLD end tag to be moved before the)MODEL statement. This is because ISPF does not allow any panel body definition after the)MODEL statement.

Conditions

- The LSTFLD tag requires an end tag.
- You must code the LSTFLD tag within an AREA, REGION, or PANEL definition. See "AREA (Area)" on page 223, "REGION (Region)" on page 445, and "PANEL (Panel)" on page 415 for descriptions of these tags.
- You can code only one list field on an application panel.

- You should code a CMDAREA on any panel that contains a LSTFLD definition. If you do not include the CMDAREA tag, the conversion utility inserts one and issues a message, unless the PANEL tag specifies CMDLINE=NO.
- You can use the SCROLLVAR attribute only once within a panel.
- The resulting scroll entry on the command line must leave at least 8 positions for the command entry field.
- If you specify the SCRVHELP attribute, you must also specify the SCROLLVAR attribute.

Nested Tags

You can code the following tags within a LSTFLD definition:

Tag	Name	Usage	Page	Required
COMMENT	Comment	Multiple	282	No
LSTCOL	List column	Multiple	372	No
LSTGRP	List group	Multiple	386	No
LSTVAR	List variable	Multiple	389	No
SOURCE	Source	Multiple	472	No

Example

The application panel in the following source file markup contains a list field with five list columns of data. In addition, three list groups are defined within the list field. The first three list columns are output-only columns. The fourth list column uses the default value *both*, which allows it to handle both input and output data. The last list column is an input-only column, and input by the user is required. Figure 139 on page 386 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=namecls TYPE='char 15'>
<VARCLASS NAME=midcls  TYPE='char 1'>
<VARCLASS NAME=phoncls TYPE='char 12'>
<VARCLASS NAME=appcls  TYPE='char 1'>
  <XLATL FORMAT=upper>
  </XLATL>
  <CHECKL>
    <CHECKI TYPE=values PARM1=EQ PARM2='Y N'>
  </CHECKL>

<VARLIST>
  <VARDCL NAME=xfname  VARCLASS=namecls>
  <VARDCL NAME=xlname  VARCLASS=namecls>
  <VARDCL NAME=xmid    VARCLASS=midcls>
  <VARDCL NAME=xphone  VARCLASS=phoncls>
  <VARDCL NAME=xapp    VARCLASS=appcls>
</VARLIST>

<PANEL NAME=lstfld3 KEYLIST=keyltbl>Subscriber List
<TOPINST>Enter phone number, if missing,
(format - nnn-nnn-nnnn) and approved
indicator (y or n) for each person.
<AREA>
  <LSTFLD SCROLLVAR=scrlamt SCRVHELP=scrhelp>
    <LSTGRP HEADLINE=yes>Subscriber Name
      <LSTCOL DATAVAR=xfname USAGE=out COLWIDTH=15>First Name
      <LSTCOL DATAVAR=xlname USAGE=out COLWIDTH=15>Last Name
      <LSTCOL DATAVAR=xmid   USAGE=out COLWIDTH=1>MI
    </LSTGRP>
```

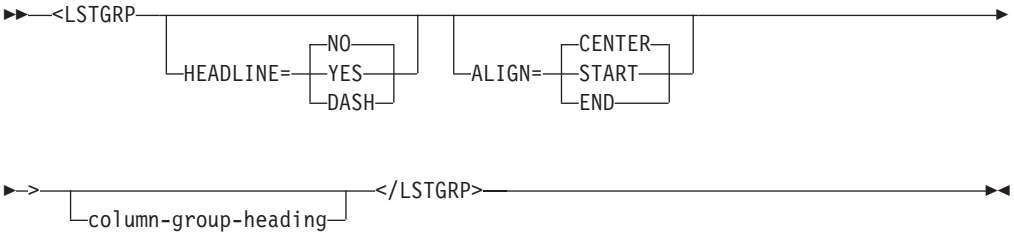
```
    <LSTGRP>Phone
      <LSTCOL DATAVAR=xphone COLWIDTH=12>Number
    </LSTGRP>
    <LSTGRP>Approved
      <LSTCOL DATAVAR=xapp USAGE=in REQUIRED=yes
        COLWIDTH=1 MSG=msgg886>(Y or N)
    </LSTGRP>
  </LSTFLD>
</AREA>
<CMDAREA>
</PANEL>
```

```
                              Subscriber List

Enter phone number, if missing, (format - nnn-nnn-nnnn) and approved
indicator (y or n) for each person.

--------- Subscriber Name ----------  Phone          Approved
First Name       Last Name        MI  Number         (Y or N)
Pete             Moss             P   919-888-4444   _
Sally            Forth            N   ____________   _
Melba            Toast            T   919-444-8888   _
****************************** BOTTOM OF DATA ****************************




Command ===> ____________________________________________ Scroll ===> ____
 F1=Help       F2=Split       F3=Exit       F7=Backward    F8=Forward
 F9=Swap       F12=Cancel
```

Figure 139. List Field

LSTGRP (List Group)

The LSTGRP tag defines a heading for a single column or multiple columns within a list field.



HEADLINE=NO | YES | DASH
This attribute specifies whether the heading text is padded to span the width of the group heading not occupied by the text. This provides a visual indication of the columns that belong to a group heading. You must specify YES or DASH to produce the visible indicator. When the GRAPHIC invocation option is specified, YES produces a solid line for host display and DASH

produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both YES and DASH produce a dashed line.

ALIGN=CENTER | START | END

This attribute specifies how the list group heading is formatted. If you do not specify this attribute, or if you specify ALIGN=CENTER, then the heading will be centered over multiple columns or a variable model line, or left-justified over a single column.

When ALIGN=START, the list group heading is left-justified. When ALIGN=END, the list group heading is right-justified.

column-group-heading
The heading is placed above the column group in the nonscrollable part of the list field. The heading must fit on one line above the column or columns in the group. If *column-group-heading* text is longer than the formatted width of the column or columns in the group, it is truncated. The *column-group-heading* appears on the line immediately above the group of columns.

If you do not specify *column-group-headings* for any of the columns within the group, the conversion utility will reserve the area where the heading would be displayed and fill it with blanks. If the *column-group-heading* is not specified but HEADLINE=YES is specified, the heading will contain only a dashed line.

Description

The LSTGRP tag defines a heading for a single column or multiple columns within a list field. You can use the LSTGRP tag to group columns in a list field together under a single heading that applies to all of the columns. You create the columns using the LSTCOL or LSTVAR tag.

The list field can contain other columns that do not belong to the list column group. Only the LSTCOL or LSTVAR definitions nested within the LSTGRP tag belong to the group.

There must be at least one LSTCOL tag, nested LSTGRP tag, or LSTVAR tag defined within a column group. The column formatting widths, and the gutters between them, define how much space is allocated for the group heading. If this space is less than the space needed for the group heading, the conversion utility will truncate the heading. If the LSTGRP definition contains only one LSTCOL tag, and the ALIGN attribute is not specified, the group heading is left-justified over the column.

You can use the LSTGRP tag to specify multiple lines of single column headings or multiple lines of multiple column headings.

Conditions

- The LSTGRP tag requires an end tag.
- You must code the LSTGRP tag within a LSTFLD definition or another LSTGRP definition. See “LSTFLD (List Field)” on page 382 for a complete description of the LSTFLD tag.
- You can code multiple LSTGRP tags within a LSTFLD definition.
- A LSTGRP definition must contain a nested LSTCOL, LSTVAR, or LSTGRP tag, otherwise the conversion utility will issue an error.
- The nested tags LSTCOL definitions must include at least one data column from the first displayable model line.

Nested Tags

You can code the following tags within the LSTGRP tag:

Tag	Name	Usage	Page	Required
COMMENT	Comment	Multiple	282	No
HP	Highlighted phrase	Multiple	354	No
LSTCOL	List column	Multiple	372	Yes
LSTGRP	List group	Multiple	386	No
LSTVAR	List variable	Multiple	389	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SOURCE	Source	Multiple	472	No

Example

In the following source file markup, the application panel contains a list field with six list columns. The first three columns are placed under a common group, as are the last two columns. Also, for each of the first three columns, a second-level group heading is used in place of list column headings. This technique provides a blank space between the group headings and the data columns. Figure 140 on page 389 shows the formatted result of the application panel.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=namecls TYPE='char 12'>
<VARCLASS NAME=midcls  TYPE='char 1'>
<VARCLASS NAME=yearcls TYPE='char 9'>
<VARCLASS NAME=semcls  TYPE='char 2'>

<VARLIST>
  <VARDCL NAME=xfname  VARCLASS=namecls>
  <VARDCL NAME=xlname  VARCLASS=namecls>
  <VARDCL NAME=xmid    VARCLASS=midcls>
  <VARDCL NAME=xyear   VARCLASS=yearcls>
  <VARDCL NAME=sem1    VARCLASS=semcls>
  <VARDCL NAME=sem2    VARCLASS=semcls>
</VARLIST>

<PANEL NAME=lstgrp WIDTH=66 KEYLIST=keyltbl>Class Roster
<AREA>
  <LSTFLD>
    <LSTGRP HEADLINE=yes>Student Name
    <LSTGRP>Last
      <LSTCOL DATAVAR=xlname USAGE=out COLWIDTH=12>
    </LSTGRP>
    <LSTGRP>First
      <LSTCOL DATAVAR=xfname USAGE=out COLWIDTH=12>
    </LSTGRP>
    <LSTGRP>M
      <LSTCOL DATAVAR=xmid   USAGE=out COLWIDTH=1>
    </LSTGRP>
    </LSTGRP>
    <LSTGRP>Class
    <LSTGRP>Year
      <LSTCOL DATAVAR=xyear USAGE=out COLWIDTH=9>
    </LSTGRP>
    </LSTGRP>
    <LSTGRP HEADLINE=yes>Grade
      <LSTCOL DATAVAR=sem1 COLWIDTH=2>Sem 1
      <LSTCOL DATAVAR=sem2 COLWIDTH=2>Sem 2
```

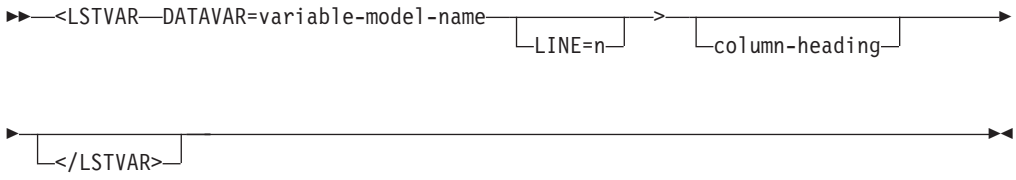
```
    </LSTGRP>
  </LSTFLD>
</AREA>
<CMDAREA>
</PANEL>
```

```
                          Class Roster           ROW 1 TO 6 OF 6

 ------- Student Name --------  Class      -- Grade ---

 Last          First         M  Year
                                             Sem 1  Sem 2
 Duff          Dean          T  Junior      A      B+
 Gillihan      Dana          L  Freshman    B+     B
 Rivas         Sergio        J  Post-Grad   D      D
 Romero        Maria         C  Post-Grad   A      A
 Spencer       Donald        M  Freshman    A      B
 Szabo         Imre          B  Senior      C+     B
 ************************ BOTTOM OF DATA ************************




Command ===> _____________________________________________________
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward
 F9=Swap     F12=Cancel
```

Figure 140. List Group

LSTVAR (List Variable)

The LSTVAR tag defines a)MODEL section variable model line displayed in the ISPF table display area of a panel.



DATAVAR=variable-model-name
This is the data which will occupy the column. The *variable-model-name* value must be a variable model line name (without a leading % sign).

LINE=n
This attribute provides the application the ability to place a LSTVAR model variable on different model lines. ISPF defines the range of lines as 1 to 8. The default is 1. Headings are generated on multiple lines to match the LSTVAR field placement.

column-heading
This is the text of the model variable heading.

Description

In conjunction with the LSTFLD and LSTCOL tags, LSTVAR tags provide a means of defining a vertically scrollable list display area that is made up of data coming from ISPF tables. One or more ISPF)MODEL section statements will be built to display the fields defined by the LSTVAR tags. The use of LSTVAR tags requires the use of the TBDISPL service in the application program.

The application must place valid data in the variable model line before the panel is displayed.

You can use the LINE attribute to format your table to display on multiple lines.

Conditions

- You must code the LSTVAR tag within a LSTFLD tag. See "LSTFLD (List Field)" on page 382 for a complete description of this tag.
- Only MODEL lines that are not blank fields are written in the panel body. Thus, if one LSTVAR entries specifies LINE=1 and another specifies LINE=3, but there are no entries for LINE 2, only two MODEL lines are created.

Nested Tags

You can code the following tags within the LSTVAR tag:

Tag	Name	Usage	Page	Required
COMMENT	Comment	Multiple	282	No
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SOURCE	Source	Multiple	472	No

Example

In the following source file markup, the application panel contains a list field with five list columns and 2 variable model lines. The first three columns are defined as output-only, and are coded within the **Subscriber Name** list group. The **Number** column is an input/output column, and it is coded within the **Phone** list group. The last column is input-only, and it is coded within the **Approved** list group. This column requires input, so if it is not filled in, the error message ***MSGG886*** will be displayed. The variable declarations and classes associated with the list columns are also shown.

Note: The variable model lines are shown in the formatted output to illustrate the formatting process. The application must provide valid values for these variables before the panel is displayed.

Figure 141 on page 391 shows the formatted result of the application panel.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=namecls TYPE='char 15'>
<VARCLASS NAME=midcls  TYPE='char 1'>
<VARCLASS NAME=phoncls TYPE='char 12'>
<VARCLASS NAME=appcls  TYPE='char 1'>
 <XLATL FORMAT=upper>
 </XLATL>
<CHECKL>
 <CHECKI TYPE=values PARM1=EQ PARM2='Y N'>
```

```
  </CHECKL>

<VARLIST>
  <VARDCL NAME=xfname  VARCLASS=namecls>
  <VARDCL NAME=xlname  VARCLASS=namecls>
  <VARDCL NAME=xmid    VARCLASS=midcls>
  <VARDCL NAME=xphone  VARCLASS=phoncls>
  <VARDCL NAME=xapp    VARCLASS=appcls>
 </VARLIST>

<PANEL NAME=lstvar KEYLIST=keyltbl>Subscriber List
<TOPINST>Enter phone number, if missing,
(format - nnn-nnn-nnnn) and approved
indicator (y or n) for each person.
<AREA>
  <LSTFLD>
    <LSTVAR datavar=xmodelv1>Variable model line at top
 <LSTGRP HEADLINE=yes>Subscriber Name
      <LSTCOL DATAVAR=xfname USAGE=out line=2 COLWIDTH=15>First Name
      <LSTCOL DATAVAR=xlname USAGE=out line=2 COLWIDTH=15>Last Name
      <LSTCOL DATAVAR=xmid   USAGE=out line=2 COLWIDTH=1>MI     </LSTGRP>
 <LSTGRP>Phone
 <LSTCOL DATAVAR=xphone line=2 COLWIDTH=12>Number
    </LSTGRP>
    <LSTGRP>Approved
 <LSTCOL DATAVAR=xapp USAGE=in line=2 REQUIRED=yes
       COLWIDTH=1 MSG=msgf886>(Y or N)
    </LSTGRP>
<LSTVAR datavar=xmodelv2 line=3>Variable model line at bottom
 </LSTFLD>
</AREA>
<CMDAREA>
</PANEL>
```

```
                              Subscriber List

 Enter phone number, if missing, (format - nnn-nnn-nnnn)&cont.
 and approved
 indicator (y or n) for each person.

 --------- Subscriber Name ----------  Phone
Approved
 Variable model line at top
 First Name       Last Name        MI  Number         (Y or N)
 Variable model line at bottom
&XMODELV1
 _______________  _______________  _   ___________    _
&XMODELV2









 Command ===> ____________________________________________________________
  F1=Help       F2=Split       F3=Exit        F7=Backward     F8=Forward
  F9=Swap      F12=Cancel
```

Figure 141. List Variable

M (Mnemonic)

The M tag defines a single character to be used as a mnemonic selection for action bar or pull-down choices.

mnemonic-character
The single-byte character following the mnemonic start tag specifies the mnemonic for the action bar choice or pull-down choice. The mnemonic-character must be a single-byte alphabetic or numeric character; double-byte characters are not allowed.

If you want the mnemonic to be a character that is not part of the normal choice text, follow the choice text with the mnemonic character specified within parenthesis. This convention is particularly useful when you have a large number of choices, which makes it difficult to choose a unique mnemonic for each choice. For example, if you had the pull-down choice Add, and the characters A and d were already used on other choices in the same pull-down, you could choose another character for your mnemonic:

```
<pdc>Add(<m>B)
```

In this case B becomes the mnemonic for Add. When an alternate mnemonic is specified on a PDC tag, the resulting 3 bytes of text (the "(B)" in this example), are replaced in the description by a variable. The variable is set to blanks or to the 3-byte value depending on the ZGUI variable, thus displaying the mnemonic only in GUI mode.

Description

Mnemonic characters are supported by ISPF for pull-down choices only when you are running in GUI mode. They are ignored for host display.

Unless you have specified MNEMGEN=NO on the AB tag, the conversion utility will automatically select a mnemonic character for each action bar and pull-down choice for SBCS conversions. The character selected as the mnemonic is the first alphabetic or numeric character in the choice description not previously used as a mnemonic for that set of choices.

Conditions

When the conversion utility automatically generates mnemonics, the M tag selection is processed first, and if the specified mnemonic is valid, the automatic mnemonic generation is not used for that choice. If the specified mnemonic character is invalid, or a duplicate of a previously used mnemonic character (either specified or automatically selected), a message is issued and an attempt will be made to select a different mnemonic character.

When processing DBCS conversions or when MNEMGEN=NO is coded on the AB tag, automatic mnemonic character selection is disabled and mnemonic characters are only specified by the M tag. The use of mnemonics should be consistent for all choices in an action bar or pull-down as follows:

- Code the M tag within the text following the ABC and PDC tags.

- Each mnemonic chosen must be unique. The conversion utility will issue a message and discard duplicate mnemonics.
- If mnemonics are used for any action bar or pull-down choice, they should be used for all of the choices. The conversion utility will issue a message if any choice in a group does not have a mnemonic.

Nested Tags

None.

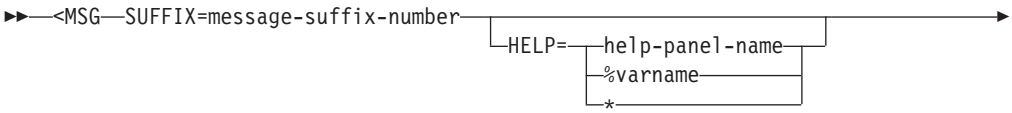
Example

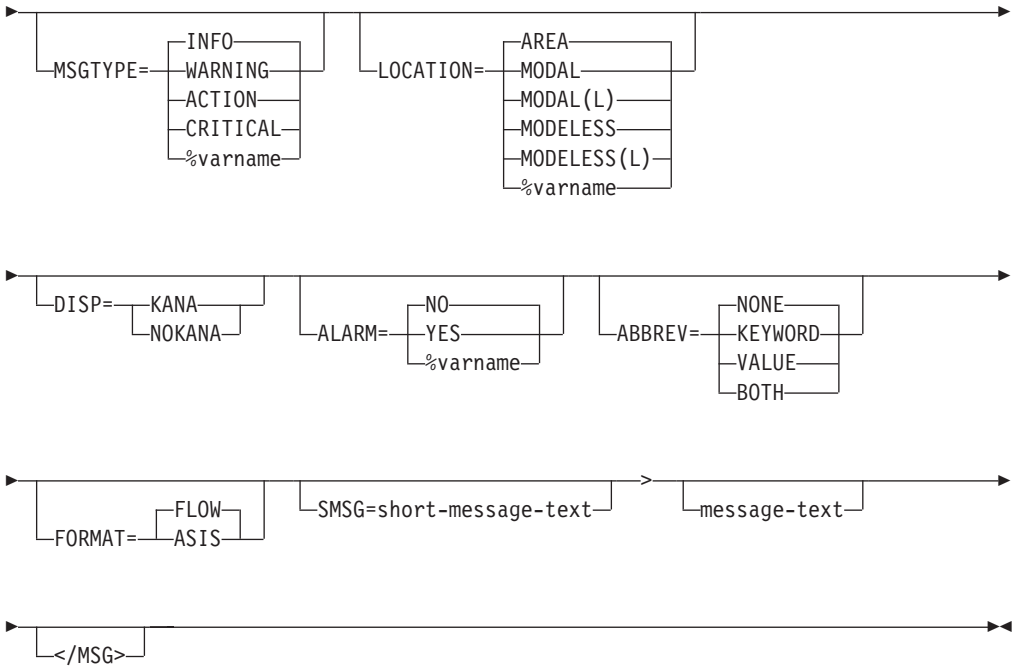
In the following example, all of the action bar choices and pull-down choices have been coded to show the use of the M tag. Some of the pull-down choices illustrate the use of the optional end tag.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 sysem>
  <!entity sampbody system>)>
&sampvar1;

<PANEL NAME=m1 KEYLIST=keylxmp>Library Card Registration
<AB>
<ABC><M>File
  <PDC><M>A</M>dd Entry
    <ACTION RUN=add>
  <PDC><M>D</M>elete Entry
    <ACTION RUN=delete>
  <PDC><M>U</M>pdate Entry
    <ACTION RUN=update>
  <PDC><M>E</M>xit
    <ACTION RUN=exit>
<ABC><M>Search
  <PDC CHECKVAR=whchsrch MATCH=1>Search on <M>n</M>ame
    <ACTION SETVAR=whchsrch VALUE=1>
    <ACTION RUN=search>
  <PDC CHECKVAR=whchsrch MATCH=2>Search on <M>c</M>ard number
    <ACTION SETVAR=whchsrch VALUE=2>
    <ACTION RUN=search>
<ABC><M>Help
  <PDC><M>Extended Help...
    <ACTION RUN=exhelp>
  <PDC><M>Keys Help...
    <ACTION RUN=keyshelp>
</AB>
&sampbody;
</PANEL>
```

MSG (Message)

The MSG tag defines a message within a message member.





SUFFIX=message-suffix-number
This attribute specifies the suffix of the message. The suffix consists of either 1 numeric character (0–9) or a numeric character (0–9) and an optional alpha suffix character as defined for ISPF messages, which is added to the MSGMBR *message-member-name* to form the ISPF message ID.

Each *message-suffix-number* within a message member must be unique. Attempts to define duplicate suffixes will result in a warning message and the duplicate MSG will be ignored.

HELP=help-panel-name | %varname | *
Specifies the name of the help panel that is associated with this message and that will be displayed if the user requests help for the message.

If you specify a help panel, ISPDTLC generates ".HELP=help-panel-name" (or ".HELP=&varname" or ".HELP=*") in the ISPF message ID definition. If you don't specify a help panel, no help is available for the message.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

MSGTYPE=INFO | WARNING | ACTION | CRITICAL | %varname
This attribute specifies the severity of the message. ISPF displays INFO messages without an alarm. ISPF displays WARNING, ACTION, and CRITICAL messages with an alarm.

ACTION and CRITICAL message types are used to identify the most severe errors. This level of error usually requires a user response. ISPF will always display CRITICAL messages in a pop-up. ACTION messages are displayed based on the value of the LOCATION attribute.

The %varname value specifies that the value INFO, WARNING, ACTION, or CRITICAL will be provided in the named variable by the application prior to issuing the message.

The conversion utility changes INFO to .TYPE=NOTIFY when formatting the message member.

ISPF recognizes message types (.TYPE=) of NOTIFY, WARNING, ACTION, and CRITICAL. ISPF uses the TYPE value specified in conjunction with the value of .WINDOW to determine the display characteristics of the message. The .WINDOW value is generated from the value specified for the LOCATION attribute. For more information on ISPF messages, refer to *ISPF Dialog Developer's Guide and Reference*

LOCATION=AREA | MODAL | MODAL(L) | MODELESS | MODELESS(L) | %varname

This attribute specifies how the message is displayed.

LOCATION=AREA (the default) specifies that the message is to appear in the panel message area. However, if the text of the message exceeds the length of the panel message area, ISPF will display the message in a pop-up.

LOCATION=MODAL specifies that the message is to appear in a pop-up which requires a user response. The conversion utility generates .WINDOW=RESP in the ISPF message definition.

LOCATION=MODAL(L) specifies that the long message is to appear in a pop-up which requires a user response. The conversion utility generates .WINDOW=LRESP in the ISPF message definition.

LOCATION=MODELESS specifies that the message is to appear in a pop-up which does not require a user response. The conversion utility generates .WINDOW=NORESP in the ISPF message definition.

LOCATION=MODELESS(L) specifies that the long message is to appear in a pop-up which does not require a user response. The conversion utility generates .WINDOW=LNORESP in the ISPF message definition.

LOCATION=%varname specifies that the value AREA, MODAL, or MODELESS will be provided in the named variable by the application prior to issuing the message. The conversion utility generates .WINDOW=&VARNAME in the ISPF message definition.

DISP=KANA | NOKANA

This attribute specifies the addition of either the KANA or NOKANA keyword to the message control information.

ALARM=NO | YES | %varname

This attribute controls the use of the alarm when the message is displayed.

ALARM=%varname specifies that the value YES or NO will be provided in the named variable by the application prior to issuing the message.

ABBREV=NONE | KEYWORD | VALUE | BOTH

This attribute specifies the format of the message control information. You may abbreviate the message control keyword, the message control keyword value, or both.

FORMAT=FLOW | ASIS

This attribute specifies the formatting of the *message-text*.

The default of **FLOW** means to flow the message text continuously within the WIDTH of the MSGMBR

When FORMAT=ASIS, the genrated message preserves imbedded blanks, but drops leading or trailing blanks.

SMSG=short-message-text
You can provide a short message of up to 24 bytes which ISPF will display in the short message area of the panel.

The VARSUB tag is not supported within the *short-message-text*. If a substitution variable is required, you may code "&variable" to place the variable name in the message. A *short-message-text* consisting of more than one word must be enclosed within quotation marks (" "). If the *short-message-text* contains a single apostrophe ('), the conversion utility generates double apostrophes as it does for *message-text*, as described below.

The short message is not recommended by the CUA Architecture definition.

A short message cannot be created unless the *message-text* is also provided.

message-text
This is the text of the message. The *message-text* is placed in the long-message area of a message file. The *message-text* is limited to 512 characters. The conversion utility truncates all *message-text* after 512 characters and issues a warning message. If no *message-text* is coded, then no message is generated.

Several characters within the long message area have a special meaning to ISPF. If you use the apostrophe within *message-text*, the conversion utility generates double apostrophes so the single apostrophe will be displayed when ISPF issues the message. If you use the ampersand (&) within the long message, it must be coded as "&amp" followed by a blank or semicolon to be interpreted as a literal ampersand character (through ENTITY substitution).

For ISPF substitution variables, you should code the VARSUB tag. ISPF does not perform output translation (specified in the associated VARCLASS tag) on ISPF run-time substitution variables.

Refer to the *ISPF Dialog Developer's Guide and Reference* for a description of the syntax rules you should use for defining consistent messages.

Description

The MSG tag defines a message within a message member. Each MSG definition within a message member must have a unique *message-suffix-number*.

Conditions

- You must code the MSG tag within a MSGMBR definition. See "MSGMBR (Message Member)" on page 397 for a complete description of this tag.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested Tags

You can code the following tag within a MSG definition:

Tag	Name	Usage	Page	Required
VARSUB	Variable substitution	Multiple	489	No

Example

The following markup contains the message member *MSGG88,* which contains nine MSG definitions. The text of messages *MSGG883* and *MSGG888* contain variable substitutions. Figure 142 shows the generated ISPF message member.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=msgcls TYPE='char 20'>
<VARLIST>
  <VARDCL NAME=phoneno VARCLASS=msgcls>
  <VARDCL NAME=cnum    VARCLASS=msgcls>
</VARLIST>

<MSGMBR NAME=msgg88>
  <MSG SUFFIX=1 disp=kana abbrev=keyword>Name must be alphabetic.
  <MSG SUFFIX=2 disp=nokana abbrev=value>Enter only number of days.
  <MSG SUFFIX=3 MSGTYPE=critical>The only rooms we have available
      are either SINGLE or DOUBLE.  Please call the manager of the hotel
      who will arrange equivalent lodging at another
      hotel in the area.  This is our mistake, and we will, of course,
      pick up the bill.  Please call collect <VARSUB VAR=phoneno>.
  <MSG SUFFIX=4 MSGTYPE=action LOCATION=modal abbrev=both>
      Please enter either BIGCHARGE, V I S T A, EZCARD, CHECK, or CASH.
  <MSG SUFFIX=5 MSGTYPE=warning LOCATION=modeless>Please enter your name.
  <MSG SUFFIX=6>Please enter Y or N.
  <MSG SUFFIX=7>Card number is a seven-digit number.
  <MSG SUFFIX=8 MSGTYPE=warning>The card number you entered,
       <VARSUB VAR=cnum> is not valid.
  <MSG SUFFIX=9>Message '9' contains imbedded quotes.
</MSGMBR>
```

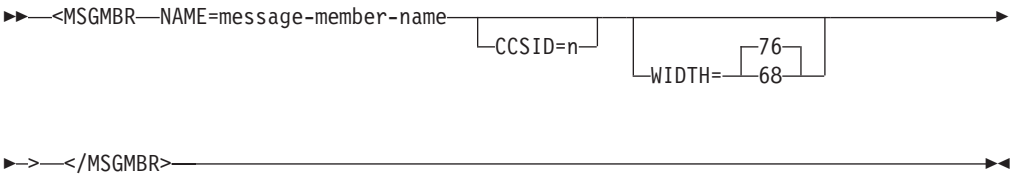
```
MSGG881 .T=NOTIFY KANA
'Name must be alphabetic.'
MSGG882 .TYPE=N NOKANA
'Enter only number of days.'
MSGG883 .TYPE=CRITICAL
'The only rooms we have available are either SINGLE or DOUBLE. Please call th' +
'e manager of the hotel who will arrange equivalent lodging at another hotel ' +
'in the area. This is our mistake, and we will, of course, pick up the bill. ' +
'Please call collect &PHONENO.'
MSGG884 .T=A .W=R
'Please enter either BIGCHARGE, V I S T A, EZCARD, CHECK, or CASH.'
MSGG885 .TYPE=WARNING .WINDOW=NORESP
'Please enter your name.'
MSGG886 .TYPE=NOTIFY
'Please enter Y or N.'
MSGG887 .TYPE=NOTIFY
'Card number is a seven-digit number.'
MSGG888 .TYPE=WARNING .ALARM=YES
'The card number you entered, &CNUM is not valid.'
MSGG889 .TYPE=NOTIFY
'Message '9'' contains imbedded quotes.'
```

Figure 142. Messages

MSGMBR (Message Member)

The MSGMBR tag defines a message member.

MSGMBR



NAME=message-member-name
This specifies the name of the message member, which also serves as the prefix for all identifiers of messages within the member.

The *message-member-name* can be specified as a 3–7 character name, conforming to ISPF message member standard naming convention. The last two positions must be numeric. The preceding characters can be A–Z, a–z, or #, $, @.

Lowercase characters are translated to their uppercase equivalents.

If you specify NAME=*, the *message-member-name* is set to the input DTL source member name. If multiple dialog element definitions have been combined within a single source file, then this notation should be used for only one dialog element definition within the file. See "Dialog Elements" on page 5 for a description of dialog element types created by the conversion utility.

The *message-member-name* is also used to build the name used for storing messages. For example, if NAME=MSGA12, the default name used to store the message members is userid.MSGS(MSGA12). This can be changed by specifying a message file on the conversion utility invocation panel. See "Chapter 10. Using the Conversion Utility" on page 175 for more information on ISPDTLC syntax.

For information on allocating a message library at run time, refer to the *ISPF User's Guide*

CCSID=n
CCSID specifies the coded-character-set identifier as defined by the Character Data Representation Architecture. CCSID should be entered as a five-position numeric value. For more information on using the CCSID attribute, refer to the *ISPF Dialog Developer's Guide and Reference*

WIDTH=76 | 68
This attribute specifies the width of the formatted messages. When WIDTH=68, the resulting messages are formatted entirely within a normal Edit or View screen.

Description

The MSGMBR tag defines a message member. You can code multiple message members for a single application.

The *message-member-name* is an explicit part of the identifier for messages coded in the message member. Each message member contains multiple messages. You use the MSG tag to define messages within a message member.

Conditions

- The MSGMBR tag requires an end tag.
- You cannot code the MSGMBR tag within any other tag definition.

Nested Tags

You can code the following tag within a MSGMBR definition:

Tag	Name	Usage	Page	Required
COMMENT	Comment	Multiple	282	No
MSG	Message	Multiple	393	Yes

Example

The following markup defines the message member *MSGM88*, which contains nine MSG definitions. Figure 143 on page 400 shows the generated ISPF message member.

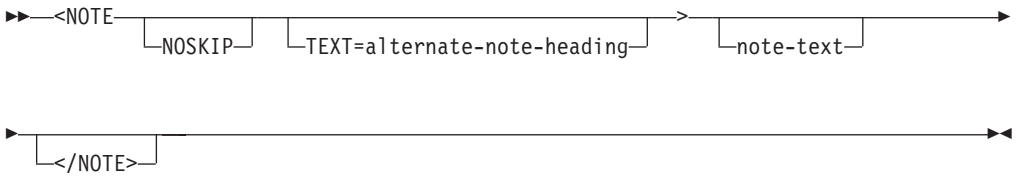
```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=msgcls TYPE='char 20'>
<VARLIST>
  <VARDCL NAME=phoneno VARCLASS=msgcls>
  <VARDCL NAME=cnum    VARCLASS=msgcls>
</VARLIST>

<MSGMBR NAME=msgm88>
  <MSG SUFFIX=1>Name must be alphabetic.
  <MSG SUFFIX=2>Enter only number of days.
  <MSG SUFFIX=3 MSGTYPE=critical>The only rooms we have available
  are either SINGLE or DOUBLE.  Please call the manager of the hotel
  who will arrange equivalent lodging at another
  hotel in the area.  This is our mistake, and we will, of course,
  pick up the bill.  Please call collect <VARSUB VAR=phoneno>.
  <MSG SUFFIX=4 MSGTYPE=action LOCATION=modal>Please enter either
  BIGCHARGE, V I S T A, EZCARD, CHECK, or CASH.
  <MSG SUFFIX=5 MSGTYPE=warning LOCATION=modeless>Please enter your name.
  <MSG SUFFIX=6>Please enter Y or N.
  <MSG SUFFIX=7>Card number is a seven-digit number.
  <MSG SUFFIX=8 MSGTYPE=warning>The card number you
  entered, <VARSUB VAR=cnum> is not valid.
  <MSG SUFFIX=9>Message '9' contains imbedded quotes.
</MSGMBR>
```

```
MSGM881 .TYPE=NOTIFY
'Name must be alphabetic.'
MSGM882 .TYPE=NOTIFY
'Enter only number of days.'
MSGM883 .TYPE=CRITICAL
'The only rooms we have available are either SINGLE or DOUBLE. Please call th' +
'e manager of the hotel who will arrange equivalent lodging at another hotel ' +
'in the area. This is our mistake, and we will, of course, pick up the bill. ' +
'Please call collect &PHONENO.'
MSGM884 .TYPE=ACTION .WINDOW=RESP
'Please enter either BIGCHARGE, V I S T A, EZCARD, CHECK, or CASH.'
MSGM885 .TYPE=WARNING .WINDOW=NORESP
'Please enter your name.'
MSGM886 .TYPE=NOTIFY
'Please enter Y or N.'
MSGM887 .TYPE=NOTIFY
'Card number is a seven-digit number.'
MSGM888 .TYPE=WARNING
'The card number you entered, &CNUM is not valid.'
MSGM889 .TYPE=NOTIFY
'Message '9'' contains imbedded quotes.'
```

Figure 143. Message Member

NOTE (Note)

The NOTE tag defines a single-paragraph note within an information region.



NOSKIP
This attribute causes the note to be formatted without creating a blank line before the note.

TEXT=alternate-note-heading
This attribute provides a text string to replace the standard ″Note:″ heading.

note-text
This is the text of the note.

Description

The NOTE tag defines a single-paragraph note within an information region. You can code the NOTE tag anywhere within an INFO tag.

The text of the note formats as an implied paragraph, at the current left margin. The text “Note:” (or its translated equivalent), or the alternate note heading, begins the paragraph and is aligned with the text of a list item when you use it within a list.

Conditions

- You must code the NOTE tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.
- You cannot nest a NOTE tag within another NOTE definition.

Nested Tags

You can code the following tags within a NOTE tag:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

The following help panel markup contains a note. Figure 144 shows the formatted result.

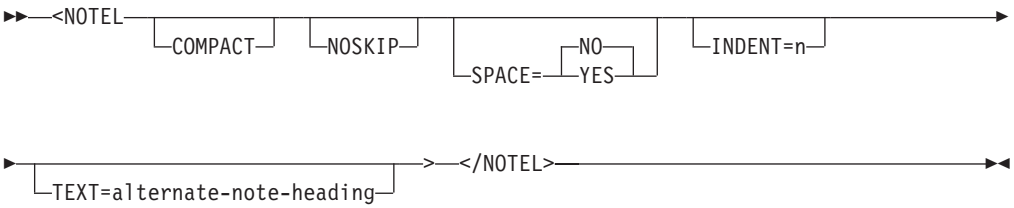
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=note DEPTH=20>Book / Periodical Search Help
<AREA>
<INFO>
  <P>This entry screen allows you to locate a desired
  book or periodical by entering the title in the entry field.
  <NOTE>If the item you are trying to locate is not
  in stock and you would like to reserve it, please see the
  librarian at the front desk.
</INFO>
</AREA>
</HELP>
```

```
          Book / Periodical Search Help

 This entry screen allows you to locate a desired
 book or periodical by entering the title in the
 entry field.

 Note: If the item you are trying to locate is
 not in stock and you would like to reserve it,
 please see the librarian at the front desk.









  F1=Help         F3=Exit         F5=Exhelp
  F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
 F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 144. Note

NOTEL (Note List)

The NOTEL tag defines a list of notes within an information region.



COMPACT
This attribute causes the list to be formatted without a blank line between the list items.

NOSKIP
This attribute causes the list to format without creating a blank line before the first line of the list.

SPACE=NO | YES
The SPACE attribute controls the indentation space for the list item. When the SPACE attribute is not specified on the LI tag, the SPACE attribute from the NOTEL tag is used to set the indentation space for the nested LI tag *item-text*.

When SPACE=YES, the indentation is set to 3 spaces.

When SPACE=NO (or SPACE is not specified), the indentation is set to 4 spaces.

The SPACE attribute can be used to control the alignment of list items when the first word of some list items is a DBCS word preceded by a shift-out character and the first word of other list items is an SBCS word.

INDENT=n
This attribute specifies that the note list be indented from the current left margin.

TEXT=alternate-note-heading
This attribute provides a text string to replace the standard ″Notes:″ heading.

Description

The NOTEL tag defines a numbered list of notes. You can code the NOTEL tag anywhere within an INFO tag.

The first line of the note list formats with the word ″Notes:″ (or its translated equivalent) or the alternate-note-heading.

Use the LI tag to denote each list item. See “LI (List Item)” on page 364 for more information on the LI tag.

Conditions

- You must code the NOTEL tag within an INFO definition. See “INFO (Information Region)” on page 356 for a complete description of this tag.
- You cannot nest a NOTEL tag within a NOTEL definition.

Nested Tags

You can code the following tags within a NOTEL definition:

Tag	Name	Usage	Page	Required
LI	List Item	Multiple	364	No
LP	List Part	Multiple	370	No

Example

The following help panel markup contains a multiple notes. Notice the numbered format for the content of the notes, which is different from the format generated with the NOTE or NT tag. A P tag is nested within the NOTEL definition to provide an additional paragraph of note text. Figure 145 shows the formatted result.

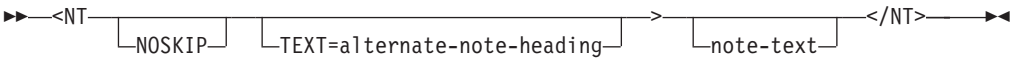
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=notel DEPTH=20>Book / Periodical Search Help
<AREA>
<INFO>
  <P>This entry screen allows you to locate a desired
  book or periodical by entering the title in the entry field.
  <NOTEL>
    <LI>If the item you are trying to locate is not
        in stock and you would like to reserve it, please see the
        librarian at the front desk.
    <LI>If the librarian is not there, please do not yell for help.
    <P>This is a library!
  </NOTEL>
</INFO>
</AREA>
</HELP>
```

```
          Book / Periodical Search Help

This entry screen allows you to locate a desired
book or periodical by entering the title in the
entry field.

Notes:

1. If the item you are trying to locate is
   not in stock and you would like to
   reserve it, please see the librarian at
   the front desk.

2. If the librarian is not there, please do
   not yell for help.

   This is a library!

  F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 145. NOTEL

NT (Note)

The NT tag defines a single- or multiple-paragraph note within an information region.



NOSKIP
This attribute causes the note to be formatted without creating a blank line before the note.

TEXT=alternate-note-heading
This attribute provides a text string to replace the standard ″Note:″ heading.

note-text
This is the text of the note. You can use the P tag to code additional paragraphs of text.

Description

The NT tag defines a single- or multiple-paragraph note within an information region. You can code the NT tag anywhere within an INFO definition.

The text of the note formats as an indented block. The block of text is indented seven spaces from the current left margin. The text “Note:” (or its translated equivalent), or the alternate note heading, begins the paragraph. The note aligns with the text of a list item when you code it within a list.

Conditions

- The NT tag requires an end tag.
- You must code the NT tag within an INFO definition. See “INFO (Information Region)” on page 356 for a complete description of this tag.
- You can nest text tags such as paragraphs and lists within a note, but you cannot nest NT and NOTE tags.

Nested Tags

You can code the following tags within a NT definition:

Tag	Name	Usage	Page	Required
DL	Definition list	Multiple	298	No
FIG	Figure	Multiple	329	No
HP	Highlighted phrase	Multiple	354	No
LINES	Lines	Multiple	366	No
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No

Tag	Name	Usage	Page	Required
XMP	Example	Multiple	497	No

Example

The following help panel markup contains a multiple-paragraph note. Notice the indented format for the content of the note, which is different from the format generated with the NOTE tag. A P tag is nested within the NT definition to provide an additional paragraph of note text. Figure 146 shows the formatted result.

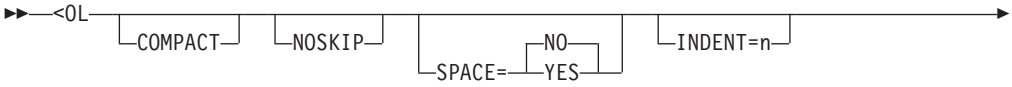
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=nt DEPTH=20>Book / Periodical Search Help
<AREA>
<INFO>
  <P>This entry screen allows you to locate a desired
  book or periodical by entering the title in the entry field.
  <NT>If the item you are trying to locate is not
  in stock and you would like to reserve it, please see the
  librarian at the front desk.
  <P>If the librarian is not there, please do not yell for help.
  This is a library!
  </NT>
</INFO>
</AREA>
</HELP>
```

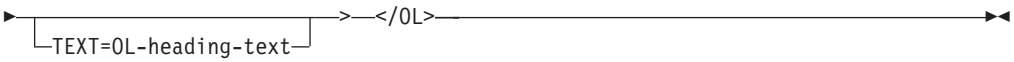
```
          Book / Periodical Search Help

 This entry screen allows you to locate a desired
 book or periodical by entering the title in the
 entry field.

 Note: If the item you are trying to locate is
       not in stock and you would like to
       reserve it, please see the librarian at
       the front desk.

       If the librarian is not there, please do
       not yell for help.  This is a library!




  F1=Help         F3=Exit         F5=Exhelp
  F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
 F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 146. NT

OL (Ordered List)

The OL tag defines an ordered list of items within an information region.





COMPACT
This attribute causes the list to be formatted without a blank line between the list items.

NOSKIP
This attribute causes the list to format without creating a blank line before the first line of the list.

SPACE=NO | YES
The SPACE attribute controls the indentation space for the list item. When the SPACE attribute is not specified on the LI tag, the SPACE attribute from the OL tag is used to set the indentation space for the nested LI tag *item-text*.

When SPACE=YES, the indentation is set to 3 spaces. When SPACE=NO (or SPACE is not specified), the indentation is set to 4 spaces.

The SPACE attribute can be used to control the alignment of list items when the first word of some list items is a DBCS word preceded by a shift-out

INDENT=n
This attribute specifies that the list be indented from the current left margin.

TEXT=OL-heading-text
This attribute causes the list to format with a heading line containing the *OL-heading-text*.

Description

The OL tag defines an ordered list of items within an information region. You use ordered lists to indicate a set of sequential items or steps. You can code the OL tag anywhere within an information region.

Ordered lists are formatted as indented lists, with sequential numbers or letters at the left margin of the list items. Nested lists (lists embedded within other lists) indent four spaces to the right of the left margin of the list that contains them.

Note: The SPACE attribute does not affect the indentation of nested lists.

The conversion utility adds a blank line before the first item in the list.

Sequential numbers or letters, depending on the nesting level of the ordered list precede the list items. The levels are as follows:
1. Level 1: 1., 2., 3., . . .
2. Level 2: a., b., c., . . .
3. Level 3: 1), 2), 3), . . .
4. Level 4: a), b), c), . . .

Any additional levels repeat the sequence from level 1.

Panels formatted with the DBCS option use uppercase alphabetic characters for the even-numbered nesting levels.

Use the LI tag to denote each list item. See “LI (List Item)” on page 364 for more information on the LI tag.

Conditions

- The OL tag requires an end tag.
- You must code the OL tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.

Nested Tags

You can code the following tags within an OL definition:

Tag	Name	Usage	Page	Required
LI	List item	Multiple	364	No
LP	List part	Multiple	370	No

Example

The following help panel markup contains two ordered lists and a paragraph. The second ordered list and the paragraph are nested within the first list. Figure 147 on page 408 shows the formatted result.

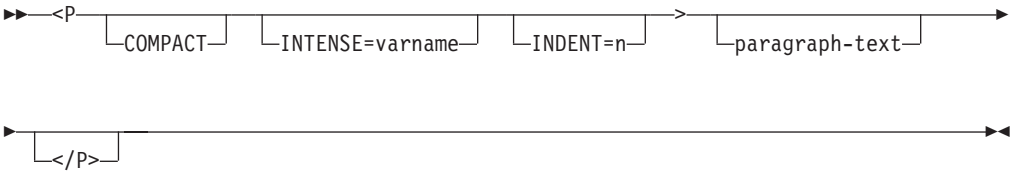
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=ol DEPTH=22 WIDTH=60>Widget Assembly Help
<AREA>
<INFO>
  <P>To assemble your new Widget, you should:
  <OL>
    <LI>Attach the gizmo flexure component to the
    main steering mechanism of the doohickey.
      <OL COMPACT>
        <LI>If slot A fits snugly on retaining
        pin B, proceed to step 2.
        <LI>If slot A does not fit snugly on
        retaining pin B, throw the Widget away
        and buy a new one.
      </OL>
    <LI>Use a screwdriver to turn the power drive unit on.
    <LI>Stand back and watch the fun!
      <P>Wake up the kids and call the neighbors, they won't
      want to miss it!
  </OL>
</INFO>
</AREA>
</HELP>
```

```
                    Widget Assembly Help

To assemble your new Widget, you should:

1.  Attach the gizmo flexure component to the main
    steering mechanism of the doohickey.

    a.  If slot A fits snugly on retaining pin B, proceed
        to step 2.
    b.  If slot A does not fit snugly on retaining pin B,
        throw the Widget away and buy a new one.

2.  Use a screwdriver to turn the power drive unit on.

3.  Stand back and watch the fun!

    Wake up the kids and call the neighbors, they won't
    want to miss it!

 F1=Help        F3=Exit        F5=Exhelp      F6=Keyshelp
 F7=PrvTopic    F8=NxtTopic   F10=PrvPage    F11=NxtPage
F12=Cancel
```

Figure 147. Ordered Lists

P (Paragraph)

The P tag defines a paragraph of text within an information region.



COMPACT
This attribute causes the paragraph to format without a blank line before the paragraph.

INTENSE=varname
This attribute supplies a variable name that must contain a valid value for the INTENS keyword. The entire paragraph is controlled by this value. For example, if the variable contains the value NON, the paragraph will not be visible.

INDENT=n
This attribute specifies that the paragraph be indented from the current left margin.

paragraph-text
This is the text of the paragraph.

Description

The P tag defines a paragraph of text within an information region. You can code the P tag anywhere within an INFO definition.

Each paragraph formats as an unindented block of text. A blank line is added before the paragraph unless the COMPACT attribute is specified.

Paragraphs within a list align with the text of the list item.

Conditions

- You must code the P tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.

Nested Tags

You can code the following tags within a P definition:

Tag	Name	Usage	Page	Required
ATTENTION	Attention	Single	232	No
CAUTION	Caution	Single	240	No
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
WARNING	Warning	Single	491	No

Example

The following help panel markup contains four paragraphs. The first three paragraphs are coded within an information region with a defined width of 40, so the text of the paragraphs will be formatted according to this width. The last paragraph is coded within an information region with no defined width, so the paragraph text will be formatted according to the width defined on the HELP tag. Figure 148 on page 410 shows the formatted result.

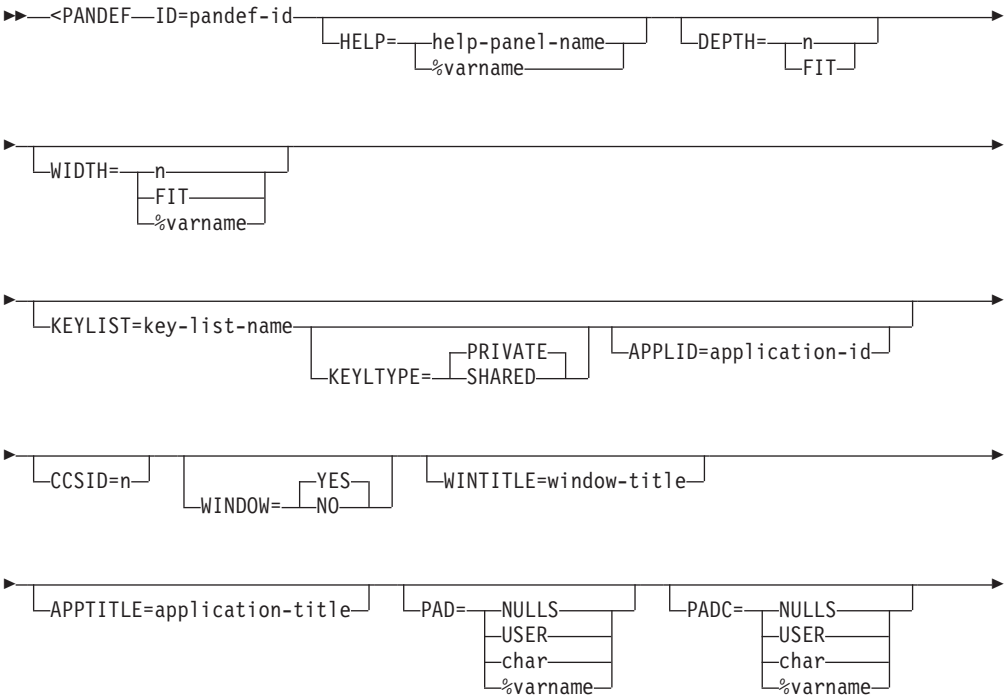
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=p DEPTH=22 WIDTH=60>P Tag Help
<AREA>
<INFO WIDTH=40>
  <P>Here's a paragraph.
  Lines are formatted to fill the width of the
  information region.
  <P>Here's another paragraph.
  Notice the line skip between the paragraphs.
  <P>Paragraphs are very versatile.
  You can use them within many other tags.
</INFO>
<INFO WIDTH=58>
  <P>The paragraphs above were formatted within an
  information region defined with a width of 40.
  This paragraph is formatted within the width specified
  for the panel.
</INFO>
</AREA>
</HELP>
```

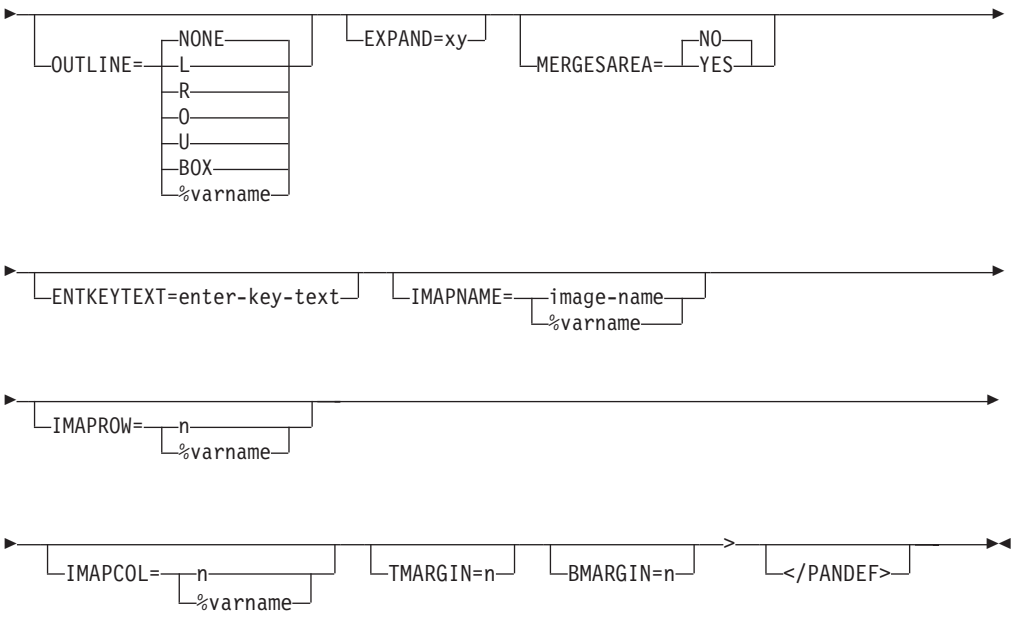
```
                        P Tag Help

Here's a paragraph. Lines are formatted
to fill the width of the information
region.

Here's another paragraph. Notice the
line skip between the paragraphs.

Paragraphs are very versatile. You can
use them within many other tags.

The paragraphs above were formatted within an information
region defined with a width of 40. This paragraph is
formatted within the width specified for the panel.




 F1=Help        F3=Exit        F5=Exhelp      F6=Keyshelp
 F7=PrvTopic    F8=NxtTopic   F10=PrvPage    F11=NxtPage
F12=Cancel
```

Figure 148. Paragraphs

PANDEF (Panel Default)

The PANDEF tag defines default values for application panels.





ID=pandef-id
This attribute defines the identifier for the panel default definition. The *pandef-id* is the value you specify with the PANDEF attribute of PANEL tags that refer to the panel default.

The *pandef-id* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

HELP=help-panel-name | %varname
This attribute specifies the extended (panel help) help panel that displays when the user selects help on an application panel that specifies the panel default.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

Specification of the HELP attribute will cause ISPDTLC to generate ".HELP=help-panel-name" (or ".HELP=&varname") in the)INIT section during panel generation.

ISPF displays this panel when the application user requests help and the cursor is not on a panel field that has its own field-level help specified. This help panel is also displayed when the user requests extended help.

DEPTH=n | FIT
This attribute specifies a default depth value for an application panel that refers to this panel default. See "PANEL (Panel)" on page 415, for more information.

WIDTH=n | FIT | %varname
This attribute specifies a default width value for an application panel that refers to this panel default. See "PANEL (Panel)" on page 415, for more information.

KEYLIST=key-list-name
This attribute specifies the name of a key mapping list associated with panels that refer to this panel default. See "KEYL (Key List)" on page 361 for more information.

KEYLTYPE= PRIVATE | SHARED
This attribute is used to add the SHARED keyword to the KEYLIST parameter of the)PANEL statement. For more information about the)PANEL statement, refer to the *ISPF Dialog Developer's Guide and Reference*

APPLID=application-id
This attribute is used to add the application ID to the)PANEL statement. The *application-id* overrides the KEYLAPPL invocation option value.

CCSID=n
This attribute specifies the default CCSID value for an application panel that refers to this panel default. See "PANEL (Panel)" on page 415 for more information.

WINDOW=YES | NO
The WINDOW attribute is used to control the generation of the WINDOW keyword on the panel)BODY section. The default is to create the WINDOW keyword. WINDOW=NO should be used when WIDTH=%varname is also used to create a panel.

WINTITLE=window-title
This attribute is used to add a title on the pop-up window border. The attribute value is placed in the ISPF ZWINTTL variable. The maximum length of the *window-title* text is the panel width minus 1.

APPTITLE=application-title
This attribute is used to add a title on the GUI window border. The attribute value is placed in the ISPF ZAPPTTL variable. The maximum length of the *application-title* text is the panel width minus 1.

PAD=NULLS | USER | char | %varname
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

PADC= NULLS | USER | char | %varname
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

OUTLINE=NONE | L | R | O | U | BOX | %varname
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

EXPAND=xy
This attribute adds the EXPAND(xy) attribute to the)BODY section of the panel. If only one character is present, the second character will be set to the same value. If the EXPAND attribute is present with no value specified, the conversion utility will use a character from the range of low-order hex values available for panel attributes. This removes an available character from possible use as a panel attribute and may cause panel formatting errors.

MERGESAREA= NO | YES
This attribute controls an additional formatting step for panels with a single scrollable area. If the entire contents of the scrollable area will fit within a standard 24-line panel (allowing 2 lines for the function keys display), and no input or output fields are found in the panel body following the location of the scrollable area, the scrollable area content is moved into the panel body.

ENTKEYTEXT=enter-key-text
This attribute is provide the text for the Enter key push button provided on

panels displayed in GUI mode. The ENTKEYTEXT attribute causes a statement to be added to the panel)INIT section to set the value of the ZENTKTXT variable to the *enter-key-text* value.

IMAPNAME=image-name | %varname
This attribute specifies the name of a image to be placed on the panel when it is displayed in GUI mode. The *image-name* is not used when the panel is displayed in host mode.

The *image-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

IMAPROW=n | %varname
This attribute specifies the row number for positioning the image. Image position uses an orgin based on 0. Therefore, the minimum row value is 0 and the maximum is 61, relating to the description for the DEPTH attribute on the PANEL tag. If a varible name is used, the application must set the variable to a valid value before the panel is displayed. The value specified should be within the actual panel depth for the image to be visible when the panel is displayed.

IMAPCOL=n | %varname
This attribute specifies the column number for positioning the image. Image position uses an orgin based on 0. Therefore, the minimum column value is 0 and the maximum is 159, relating to the description for the WIDTH attribute on the PANEL tag. If a varible name is used, the application must set the variable to a valid value before the panel is displayed. The value specified should be within the actual panel width for the image to be visible when the panel is displayed.

TMARGIN=n
This attribute provides the number of blank lines to format at the top of the panel as a top margin.

BMARGIN=n
This attribute provides the number of blank lines to format at the bottom of the panel as a bottom margin.

Description

The PANDEF tag defines default values for application panels.

PANEL tags refer to the panel default by specifying the *pandef-id* definition as the PANDEF attribute value. When a PANEL tag refers to a panel default, the values specified by the associated PANDEF tag are used for the panel unless overridden by values specified in the PANEL tag definition.

The PANEL tag can override any of the PANDEF values by specifying that value within its own definition. Thus, it is possible for a PANEL tag to select certain default values from the panel default and override others.

See "PANEL (Panel)" on page 415 for more information.

You can code multiple panel defaults for an application. Each panel default should have a unique *pandef-id*.

Conditions

- You cannot code the PANDEF tag within any other tag definition.
- You must code the PANDEF tag before you code any PANEL tag that refers to it.

- If both PAD and PADC have been specified, PAD is ignored and PADC is used.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.
- EXPAND can operate only when there are no trailing attributes on the line to be expanded. Panel lines formatted as part of a horizontal region require the use of attributes for field alignment. Therefore, the EXPAND feature is functional only for panel sections built within a vertical (or default) region that is not part of any horizontal region.

Nested Tags

None.

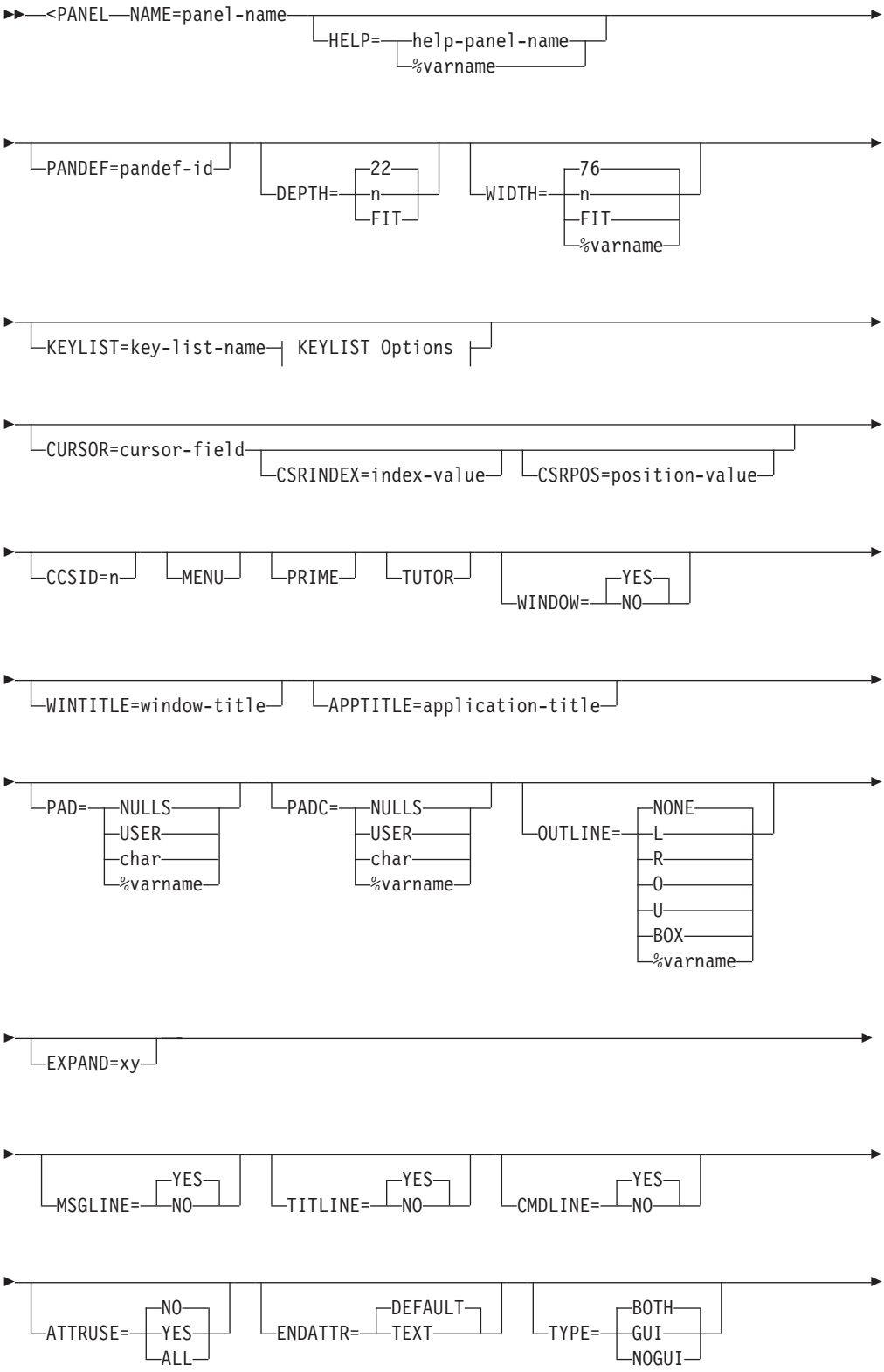
Example

The following source file markup contains two panel default definitions. The application panels *panel1* and *panel2* both refer to the panel default *pandef1*. The panel *panel1* uses all of the defined default values and *panel2* uses only the default DEPTH and WIDTH values, and overrides the default HELP and KEYLIST values by specifying those values in the PANEL definition. The third application panel, *panel3* refers to all of the default values specified in the panel default *pandef2*.

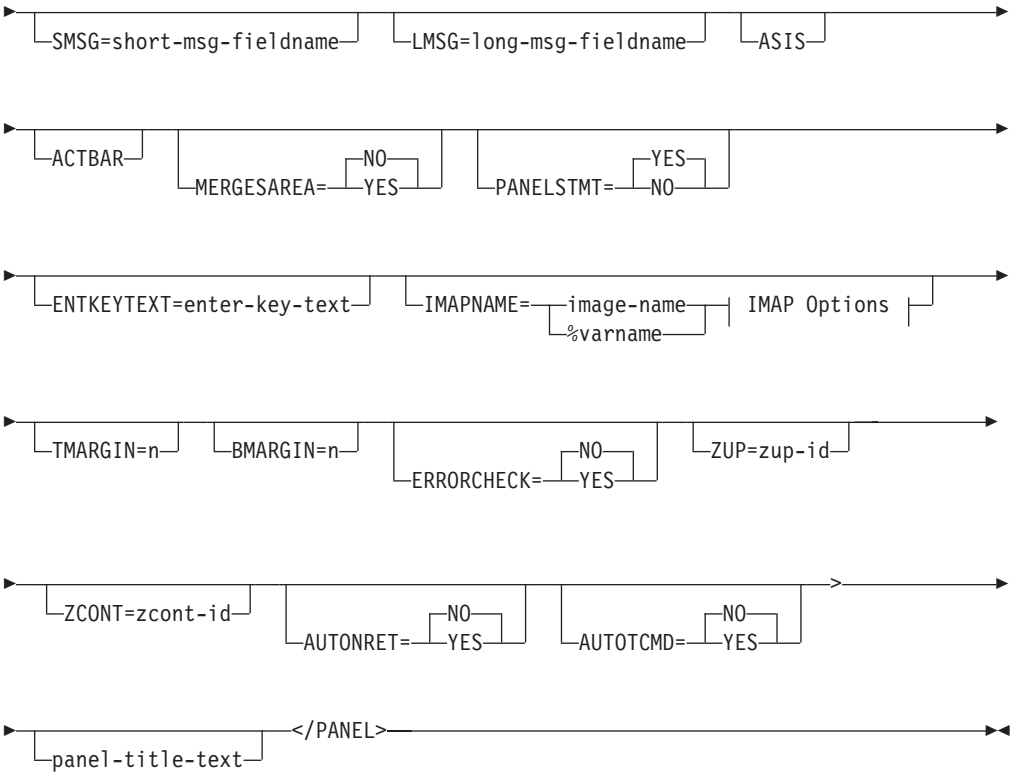
```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 sysem>
  <!entity sampbody system>)>
&sampvar1;

<PANDEF ID=panldef1 DEPTH=20 WIDTH=76 HELP=helpaaa KEYLIST=keylxmp>

<PANDEF ID=panldef2 DEPTH=22 WIDTH=70 HELP=morehlp>

<PANEL NAME=pandef1 PANDEF=panldef1>First Panel
&sampbody;
</PANEL>

<PANEL NAME=pandef2 PANDEF=panldef1
 HELP=morehlp KEYLIST=keyltbl>Second Panel
&sampbody;
</PANEL>

<PANEL NAME=pandef3 PANDEF=panldef2>Third Panel
&sampbody;
</PANEL>

<HELP NAME=helpaaa>Help panel "helpaaa"
<AREA>
<INFO WIDTH=48>
<P>This is PANDEF help panel "helpaaa"
</INFO>
</AREA>
</HELP>

<HELP NAME=morehlp>Help panel "morehlp"
<AREA>
<INFO WIDTH=48>
<P>This is PANDEF help panel "morehlp"
</INFO>
</AREA>
</HELP>
```

PANEL (Panel)

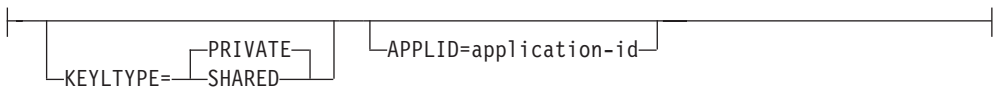
The PANEL tag defines an application panel.



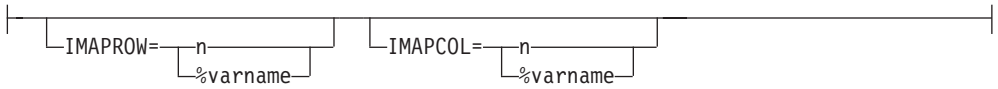
PANEL



KEYLIST Options:



IMAP Options:



NAME=panel-name

This attribute specifies the name of the panel. The *panel-name* is used in the ISPF DISPLAY or TBDISPL service call. The *panel-name* is also used as the panel ID, which the user can display. The *panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

If you specify NAME=*, the *panel-name* is set to the input DTL source member name. If multiple dialog element definitions have been combined within a single source file, then this notation should be used for only one dialog element definition within the file. See "Dialog Elements" on page 5 for a description of dialog element types created by the conversion utility.

The *panel-name* is used to build the panel output file name in which the conversion utility stores the converted panel. The default is "userid.PANELS(*panel-name*)".

You can specify the output panel library filename of your choice on the invocation panel for the conversion utility, or in the conversion utility profile as DDname DTLPAN for batch (or command syntax invocation) processing.

If the SCRIPT option has been specified, the *panel-name* is also used to build the file name in which the conversion utility stores the image of the panel. The default name is "userid.SCRIPT(*panel-name*)".

You can specify the output SCRIPT library filename of your choice on the invocation panel for the conversion utility, or in the conversion utility profile as DDname DTLSCR for batch (or command syntax invocation) processing.

See "Chapter 10. Using the Conversion Utility" on page 175 for complete information on invocation syntax.

HELP=help-panel-name | %varname
This attribute specifies the name of a defined extended (panel help) help panel. It identifies the help text that is associated with the panel definition.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

Specification of the HELP attribute will cause ISPDTLC to generate ".HELP=help-panel-name" (or ".HELP=&varname") in the)INIT section during panel generation.

ISPF displays this panel when the application user requests help and the cursor is not on a panel field that has its own field-level help specified. This help panel is also displayed when the user requests extended help.

PANDEF=pandef-id
This attribute specifies a defined panel default. The *pandef-id* value is the identifier specified on the PANDEF tag. You can override any of the defaults from this PANDEF tag by specifying that attribute on the PANEL tag. See "PANDEF (Panel Default)" on page 410 for information on defining panel defaults.

DEPTH=22 | n | FIT
This attribute defines the depth of the panel. The default depth is 22 when WINDOW=YES or 24 when WINDOW=NO. When the panel is displayed in a pop-up, ISPF adds two lines to the DEPTH value you specify to accommodate the borders at the top and bottom of the pop-up.

The value specified for the depth is the depth of the entire panel including the panel title, the action bar, the function key area, the message area, any scrollable areas, and the command area.

The maximum depth is 62 and the minimum depth is 5. If the DEPTH value is less than the minimum value allowed or exceeds the maximum value allowed, the conversion utility issues a warning message and sets the depth to the default.

The depth defined should be large enough to include all formatted text and input/output fields as well as the function key area, message area, any scrollable areas, and the command area. If the depth specified is not large enough to include these panel elements, ISPF will overlay with the function keys if the function key display is on or with the message area if the message is not displayed in a pop-up.

If DEPTH=FIT, The conversion utility will format the panel using a depth of 22. When formatting is completed the DEPTH value will be reset to the minimum depth used or to 5 if the formatted panel contains less than 5 lines.

If the DEPTH value exceeds the maximum allowed to display the panel on the device, ISPF issues an error message at run time.

WIDTH=76 | n | FIT | %varname
This attribute defines the width (in characters) of the panel. The default width is 76 when WINDOW=YES or 80 when WINDOW=NO. When the panel is displayed in a pop-up, ISPF adds 4 to the WIDTH value you specify to accommodate the left and right borders of the pop-up.

The value specified for the width is the width of the entire panel (or region), including the margins.

The maximum width is 160 and the minimum width is 16.

Because there is a minimum margin width of 1 character on each side of the panel text, the effective width for text for a panel defined with WIDTH=76 is a maximum of 74 characters.

If the WIDTH value is less than the minimum value allowed or exceeds the maximum value allowed, ISPDTLC issues a warning message and sets the width to the default.

If WIDTH=FIT or WIDTH=%varname, the conversion utility will format the panel using the maximum available width as determined from the LRECL value of the output panel file.

If WIDTH=FIT, when formatting is completed the WIDTH value will be reset to the minimum width used or to 16 if the formatted panel is less than 16 characters wide.

If WIDTH=%varname, when formatting is completed the WIDTH keyword on the)BODY panel statement is set to the variable name. WINDOW=NO must also be coded on the PANEL tag in order to use *%varname*.

Note: Panels that have the width specified as a variable cannot be preprocessed.

If WIDTH value exceeds the maximum allowed to display the panel on the device, ISPF issues an error message at run time.

KEYLIST=key-list-name
This attribute specifies the name of the key mapping list associated with the panel.

If you do not specify a *key-list-name* in a PANEL definition or an associated PANDEF definition, the ISPF-provided key list (ISPKYLST) will be used. For information on defining key mapping lists, see "KEYL (Key List)" on page 361. For information on the ISPF-provided key list, refer to the *ISPF User's Guide*

KEYLTYPE=PRIVATE | SHARED
This attribute is used to add the SHARED keyword to the KEYLIST parameter of the)PANEL statement. For information about the)PANEL statement, refer to the *ISPF Dialog Developer's Guide and Reference* The KEYLTYPE attribute is ignored if you have not provided the KEYLIST attribute as part of the PANEL tag definition or as part of an associated PANDEF tag definition.

APPLID=application-id
This attribute is used to add the application ID to the)PANEL statement. The *application-id* overrides the KEYLAPPL invocation option value. The

APPLID attribute is ignored if you have not provided the KEYLIST attribute as part of the PANEL tag definition or as part of an associated PANDEF tag definition.

CURSOR=cursor-field
This attribute, together with CSRINDEX and CSRPOS, controls the initial placement of the cursor when the ISPF displays the panel. You can specify *cursor-field* as the value of:
- The NAME attribute of a CHOICE tag (for multiple-choice selection fields)
- The DATAVAR attribute of the CHOFLD tag.
- The DATAVAR attribute of a DTAFLD tag
- The DATAVAR attribute of a LSTCOL tag
- The NAME attribute of a SELFLD tag (for single-choice selection fields).

The cursor can also be placed on the command area, when it is defined for a panel with the CMDAREA tag. Use the ISPF-reserved name *cmdarea* as the value for *cursor-field* to place the cursor on the command area.

CSRINDEX=index-value
This attribute, together with CURSOR and CSRPOS, controls the placement of the cursor when ISPF displays a table display panel. This attribute may be specified only when the CURSOR attribute refers to a list column.

CSRINDEX specifies the row in the)MODEL section where ISPF places the cursor when it displays the panel.

CSRPOS=position-value
This attribute, together with CURSOR and CSRINDEX, controls the placement of the cursor when ISPF displays the panel. This attribute may be specified only when the CURSOR attribute refers to a data field, list column, or the command area.

CSRPOS specifies the number of byte positions into the entry field that ISPF places the cursor when it displays the panel.

The first position of a field is denoted by 1. The maximum position that you can specify is the length of the underlying data.

If the value specified for this attribute is not valid, the default (1) is used.

CCSID=n
This attribute specifies the coded-character-set identifier as defined by the Character Data Representation Architecture. CCSID should be entered as a five-position numeric value. For more information on using the CCSID attribute, refer to the *ISPF Dialog Developer's Guide and Reference*

MENU
This attribute specifies that the panel will be an ISPF menu selection or edit model selection panel. This type of panel does not allow a table display.

PRIME
This attribute is used together with MENU to specify a primary option menu.

TUTOR
This attribute specifies that the panel title be formatted with the word *Tutorial* (or its translated equivalent) on each end of the title line, similar to ISPF tutorial panels.

WINDOW=YES | NO
The WINDOW attribute is used to control the generation of the WINDOW

keyword on the panel)BODY section. The default is to create the WINDOW keyword. WINDOW=NO should be used when WIDTH=%varname is also used to create a panel.

WINTITLE=window-title
This attribute is used to add a title on the pop-up window border. The attribute value is placed in the ISPF ZWINTTL variable. The maximum length of the *window-title* text is the panel width minus 1.

APPTITLE=application-title
This attribute is used to add a title on the GUI window border. The attribute value is placed in the ISPF ZAPPTTL variable. The maximum length of the *application-title* text is the panel width minus 1.

PAD=NULLS | USER | char | %varname
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

PADC= NULLS | USER | char | %varname
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

OUTLINE=NONE | L | R | O | U | BOX | %varname
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

EXPAND=xy
This attribute adds the EXPAND(xy) attribute to the)BODY section of the panel. If only one character is present, the second character will be set to the same value. If the EXPAND attribute is present with no value specified, the conversion utility will use a character from the range of low-order hex values available for panel attributes. This removes an available character from possible use as a panel attribute and may cause panel formatting errors.

MSGLINE=YES | NO
This attribute controls the provision for a long message line in the generated panel. When MSGLINE=NO, the blank line for the long message is not added to the panel)BODY section. It is the panel designer's responsibility to ensure that critical panel areas are positioned so that the long message will not inhibit use of the resulting panel.

TITLINE=YES | NO
This attribute controls the generation of the panel title line. When TITLINE=NO, the panel title is not added to the generated panel. This option is provided for applications that format a panel title as part of a dynamic area. It is the panel designer's responsibility to ensure that the resulting panel meets CUA requirements.

CMDLINE=YES | NO
This attribute controls the automatic generation of the command area on option menu panels and table display panels. When CMDLINE=NO, the command area is not automatically added to panels that do not include a CMDAREA tag within the panel definition.

ATTRUSE=NO | YES | ALL
This attribute controls the assignment of panel attributes within the range of x'01' through x'3F'. When ATTRUSE=YES or ATTRUSE=ALL, attributes for use in dynamic areas supplied by the ATTR tag can be assigned low-order hex values normally used by the conversion utility.

When ATTRUSE=YES, all of the attributes specified by the ATTR tag plus the required attributes used by the conversion utility must fit in the defined range of x'01' through x'2F'.

When ATTRUSE=ALL, all of the attributes specified by the ATTR tag plus the required attributes used by the conversion utility must fit in the defined range of x'01' through x'3F'.

ENDATTR=DEFAULT | TEXT
This attribute specifies that when the last attribute on any panel body line is "normal text" (CUA), it will be replaced by the default "text" (ISPF) attribute. The effect is to force any text on subsequent lines not preceded by another attribute from the normal text color to blue.

TYPE=BOTH | GUI | NOGUI
This attribute specifies that the panel will be used for either host display, GUI display, or both. When NOGUI is specified, for example, the panel language control statements that enable check boxes, radio buttons, list boxes, drop-down lists, and combination boxes are not added to the generated panel. When GUI is specified, SELFLD tag formatting for list boxes, drop-down lists, and combination boxes results in only 1 line in the panel)BODY section; the choice list is displayed as a GUI function.

SMSG=short-msg-fieldname
This attribute provides the name of the field where the short message is to be placed. The *short-msg-fieldname* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

LMSG=long-msg-fieldname
This attribute provides the name of the field where the long message is to be placed. The *long-msg-fieldname* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

ASIS
This attribute specifies that the command and long message fields are to appear on the display as specified in the generated panel definition. When ASIS is specified, any user request specified on the Settings panel, or by setting the system variable ZPLACE is ignored.

ACTBAR
This attribute causes the action bar information for the panel to be generated, overriding the NOACTBAR invocation option.

MERGESAREA=NO | YES
This attribute controls an additional formatting step for panels with a single scrollable area. If the entire contents of the scrollable area will fit within a standard 24-line panel (allowing 2 lines for the function keys display), and no input or output fields are found in the panel body following the location of the scrollable area, the scrollable area content is moved into the panel body.

PANELSTMT=YES | NO
This attribute controls the creation of the)PANEL statement. You can use this attribute to create a panel without keylist interaction.

ENTKEYTEXT=enter-key-text
This attribute is provide the text for the Enter key push button provided on panels displayed in GUI mode. The ENTKEYTEXT attribute causes a statement to be added to the panel)INIT section to set the value of the ZENTKTXT variable to the *enter-key-text* value.

IMAPNAME=image-name | %varname
This attribute specifies the name of a image to be placed on the panel when it is displayed in GUI mode. The *image-name* is not used when the panel is displayed in host mode.

The *image-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

IMAPROW=n | %varname
This attribute specifies the row number for positioning the image. Image position uses an orgin based on 0. Therefore, the minimum row value is 0 and the maximum is 61, based on the above description for the DEPTH attribute. If a varible name is used, the application must set the variable to a valid value before the panel is displayed. The value specified should be within the actual panel depth for the image to be visible when the panel is displayed.

IMAPCOL=n | %varname
This attribute specifies the column number for positioning the image. Image position uses an orgin based on 0. Therefore, the minimum row value is 0 and the maximum is 159, based on the above description for the WIDTH attribute. If a varible name is used, the application must set the variable to a valid value before the panel is displayed. The value specified should be within the actual panel width for the image to be visible when the panel is displayed.

TMARGIN=n
This attribute provides the number of blank lines to format at the top of the panel as a top margin.

BMARGIN=n
This attribute provides the number of blank lines to format at the bottom of the panel as a bottom margin.

ERRORCHECK=NO | YES
This attribute specifies whether error checking logic is added to the end of the)PROC section. The extra logic prevents exit from the panel if any errors are present.

```
IF (.MSG ¬= ' ')
   &ZVERB = ' '
   .RESP = ENTER
```

ZUP=zup-id
This attribute provides the name of the Tutorial panel to be assigned to the ZUP variable. It is valid only when the TUTOR attribute has been specified.

ZCONT=zcontid
This attribute provides the name of the Tutorial panel to be assigned to the ZCONT variable. It is valid only when the TUTOR attribute has been specified.

AUTONRET=NO | YES
This attribute specifies whether the .NRET = OFF panel statement is added to the)PROC section as part of the AUTOTYPE logic. When YES is specified, '.NRET = OFF' is the first AUTOTYPE panel logic statement created in the)PROC section.

AUTOTCMD=NO | YES
This attribute specifies whether the command field name is included in with the AUTOTYPE variables added to the REFRESH statement in the)REINIT section of the panel. When YES is specified the command field name (normally ZCMD) is added to the REFRESH list.

panel-title-text
This is the text of the panel title.

Panel titles should be used when more than one panel can be displayed within an application. The *panel-title-text* is centered within the width defined for the panel in accordance with CUA rules. If the *panel-title-text* is wider than the WIDTH specified, the title is truncated, from the right, with an ellipsis (...) appended. Two lines are reserved for the panel title and for a blank line between the panel title and the rest of the panel body.

Description

The PANEL tag defines an application panel.

Tags coded within a PANEL definition (between the PANEL start tag and end tag) define the content of the panel.

Conditions

- When the MENU attribute is specified, the LSTFLD tag cannot be nested under the PANEL tag.
- The PANEL tag requires an end tag.
- You cannot code a PANEL tag within any other tag definition.
- The PANEL definition must contain at least one of the following tags:
  - BOTINST (See "BOTINST (Bottom Instruction)" on page 238)
  - DA (See "DA (Dynamic Area)" on page 287)
  - DTAFLD (See "DTAFLD (Data Field)" on page 312)
  - GA (See "GA (Graphic Area)" on page 333)
  - INFO (See "INFO (Information Region)" on page 356)
  - LSTFLD (See "LSTFLD (List Field)" on page 382)
  - PNLINST (See "PNLINST (Panel Instruction)" on page 437)
  - SELFLD (See "SELFLD (Selection Field)" on page 455)
  - TOPINST (See "TOPINST (Top Instruction)" on page 477)
- If both PAD and PADC have been specified, PAD is ignored and PADC is used.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.
- EXPAND can operate only when there are no trailing attributes on the line to be expanded. Panel lines formatted as part of a horizontal region require the use of attributes for field alignment. Therefore, the EXPAND feature is functional only for panel sections built with a vertical (or default) region that is not part of any horizontal region.

Nested Tags

You can code the following tags within a PANEL definition:

Tag	Name	Usage	Page	Required
AB	Action bar	Single	211	No
AREA	Area	Multiple	223	No
BOTINST	Bottom instruction	Multiple	238	No
CMDAREA	Command area	Single	273	No
COMMENT	Comment	Multiple	282	No
DA	Dynamic area	Multiple	287	No

Tag	Name	Usage	Page	Required
DIVIDER	Area divider	Multiple	296	No
DTACOL	Data column	Multiple	306	No
DTAFLD	Data field	Multiple	312	No
GA	Graphic area	Single	333	No
GENERATE	Generate	Multiple	335	No
GRPHDR	Group header	Multiple	337	No
HP	Highlighted phrase	Multiple	354	No
INFO	Information region	Multiple	356	No
LSTFLD *	List field	Single	382	No
PNLINST	Panel Instruction	Multiple	437	No
REGION	Region	Multiple	445	No
SELFLD	Selection field	Multiple	455	No
SOURCE	Source	Multiple	472	No
TEXTLINE	Text Line	Single	475	No
TOPINST	Top instruction	Multiple	477	No

Note:
* This tag is not valid within an ISPF selection menu panel.

Example

The following application panel markup contains an action bar, a top instruction, two selection fields, and a command area. The PANEL KEYLIST attribute specifies a key mapping list, which is displayed below the command area. Figure 149 on page 425 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=selcls TYPE='CHAR 2'>
<VARLIST>
  <VARDCL NAME=loc  VARCLASS=selcls>
  <VARDCL NAME=mode VARCLASS=selcls>
</VARLIST>

<PANEL NAME=panel HELP=trvlhlp KEYLIST=keylxmp
  DEPTH=22 WIDTH=60>Dream Vacation Guide
<AB>
  <ABC>File
    <PDC>Add Entry
        <ACTION RUN=add>
    <PDC>Delete Entry
        <ACTION RUN=delete>
    <PDC>Update Entry
        <ACTION RUN=update>
    <PDC>Exit
        <ACTION RUN=exit>
  <ABC>Help
    <PDC>Extended Help...
        <ACTION RUN=exhelp>
    <PDC>Keys Help...
        <ACTION RUN=keyshelp>
</AB>
<TOPINST>Choose one of the following exotic locations and
your preferred mode of travel, then press Enter.
<AREA>
  <REGION DIR=horiz>
```

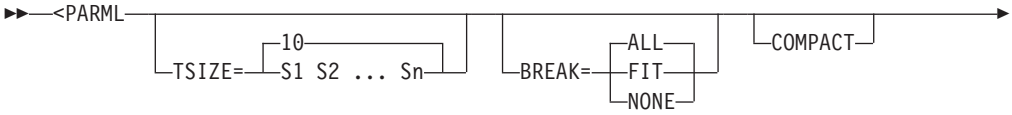
```
  <SELFLD NAME=loc PMTWIDTH=23 SELWIDTH=25>Exotic Location:
    <CHOICE>Athens, GA
    <CHOICE>Berlin, CT
    <CHOICE>Cairo, IL
    <CHOICE>Lizard Lick, NC
    <CHOICE>Paris, TX
    <CHOICE>Rome, NY
    <CHOICE>Venice, FL
  </SELFLD>
  <DIVIDER>
  <SELFLD NAME=mode PMTWIDTH=25 SELWIDTH=25>Travel Mode:
    <CHOICE>Boxcar
    <CHOICE>Hitchhike
    <CHOICE>Mule
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>
</PANEL>

<HELP NAME=trvlhlp>Sample help panel "trvlhelp"
<AREA>
<INFO WIDTH=48>
<P>This is help panel "trvlhlp"
</INFO>
</AREA>
</HELP>
```

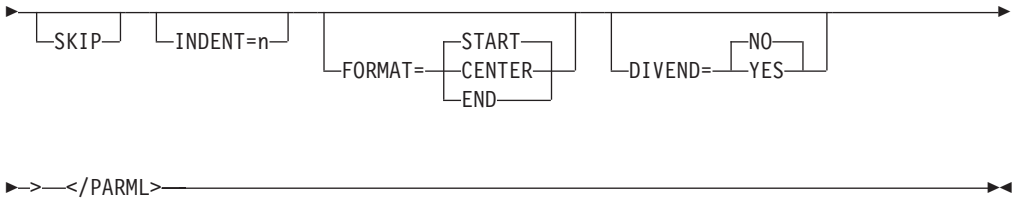
```
  File  Help
---------------------------------------------------------
                   Dream Vacation Guide

Choose one of the following exotic locations and your
preferred mode of travel, then press Enter.

Exotic Location:            Travel Mode:
__  1.  Athens, GA          __  1.  Boxcar
    2.  Berlin, CT              2.  Hitchhike
    3.  Cairo, IL               3.  Mule
    4.  Lizard Lick, NC
    5.  Paris, TX
    6.  Rome, NY
    7.  Venice, FL




Command ===> _____________________________________________
 F1=Help     F2=Split      F3=Exit     F6=Keyshelp
 F9=Swap    F12=Cancel
```

Figure 149. Application Panel

PARML (Parameter List)

The PARML tag defines a parameter list within an information region.





TSIZE=10 | S1 S2... Sn
This attribute defines the space allocated for the parameter term. The default is 10 characters. The minimum TSIZE value is 0 and the maximum is 40.

When multiple TSIZE values are specified, a PT tag must be coded for each value. The sizes are applied to the PT tags in the order the tags are encountered in the DTL source file.

BREAK=ALL | FIT | NONE
This attribute controls the formatting of the parameter terms and descriptions. If BREAK=ALL (the default), every description is on the line below the term. If BREAK=FIT, the description is on the line below the term if the term is longer than the TSIZE value. If BREAK=NONE, the term is on the same line as the description, spilling into the description area if the length exceeds the TSIZE value.

COMPACT
This attribute causes the conversion utility to format the list without a blank line between the items.

SKIP
This attribute causes a blank line to be formatted before the first parameter term when COMPACT is also specified.

INDENT=n
This attribute specifies that the parameter list be indented from the current left margin.

FORMAT=START | CENTER | END
This attribute specifies the placement of the PT tag text within the space specified by TSIZE. The PARML tag FORMAT setting applies to all of the PT tags within the parameter list.

DIVEND=NO | YES
This attribute specifies whether a divider character will be formatted following the PD tag text. When DIVEND=YES, the formatting width of the PD text is reduced to allow space for the divider character.

Description

The PARML tag defines a parameter list within an information region.

Parameter lists are similar to definition lists. They involve three tags: PARML (parameter list) and a matching end tag, PT (parameter term), and PD (parameter description). As in definition lists, the term tag defines a term, and the definition tag defines the description associated with the term. The PD tag must immediately follow the PT tag that it is associated with.

Parameter lists can occur anywhere in an information region; you can nest them within other lists, and you can nest other lists within parameter lists.

Conditions

- The PARML tag requires an end tag.
- You must code the PARML tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.

Nested Tags

You can code the following tags within a PARML definition:

Tag	Name	Usage	Page	Required
PD	Parameter description	Multiple	428	No
PLDIV	Parameter List Divider	Multiple	435	No
PT	Parameter term	Multiple	442	No
PTDIV	Paramater Term Divider	Multiple	444	No

Example

The following help panel markup contains two parameter lists. The second parameter list is nested within the second parameter description of the first list. Figure 150 on page 428 shows the formatted result.

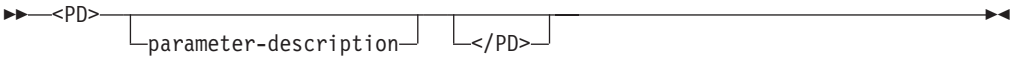
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=parmls DEPTH=22>Part Number Code Help
<AREA>
<INFO>
  <P>Valid part numbers consist of a three-digit
  number followed by a 2-character suffix.
  <PARML TSIZE=6>
    <PT>123
    <PD>The first three digits represent
    the lot number of the part.
    <PT>AA
    <PD>The 2-character suffix represents the
    department the part originated from.
    The valid suffixes are:
      <PARML BREAK=none COMPACT>
        <PT>TO
        <PD>Tools
        <PT>EL
        <PD>Electrical
        <PT>ME
        <PD>Mechanical
      </PARML>
  </PARML>
</INFO>
</AREA>
</HELP>
```

```
             Part Number Code Help

Valid part numbers consist of a three-digit
number followed by a 2-character suffix.

123
      The first three digits represent the lot
      number of the part.

AA
      The 2-character suffix represents the
      department the part originated from. The
      valid suffixes are:

      TO        Tools
      EL        Electrical
      ME        Mechanical


 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 150. Parameter List

PD (Parameter Description)

The PD tag defines a parameter description in a parameter list.



parameter-description
This is the text of the parameter description.

Description

The PD tag defines a parameter description in a parameter list.

Conditions

- You must code the PD tag within a PARML definition. See "PARML (Parameter List)" on page 425 for a complete description of this tag.
- Each PD tag must be paired with a PT tag. You can specify only one PD tag for each PT tag within a parameter list. The PD tag must immediately follow the PT tag it is associated with.

Nested Tags

You can code the following tags within a PD definition:

Tag	Name	Usage	Page	Required
DL	Definition list	Multiple	298	No
FIG	Figure	Multiple	329	No
HP	Highlighted phrase	Multiple	354	No

Tag	Name	Usage	Page	Required
LINES	Lines	Multiple	366	No
NOTE	Note	Multiple	400	No
NOTEL	Note List	Multiple	402	No
NT	Note	Multiple	404	No
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No
XMP	Example	Multiple	497	No

Example

The following help panel markup contains a parameter list with three PD definitions. Figure 151 on page 430 shows the formatted result.

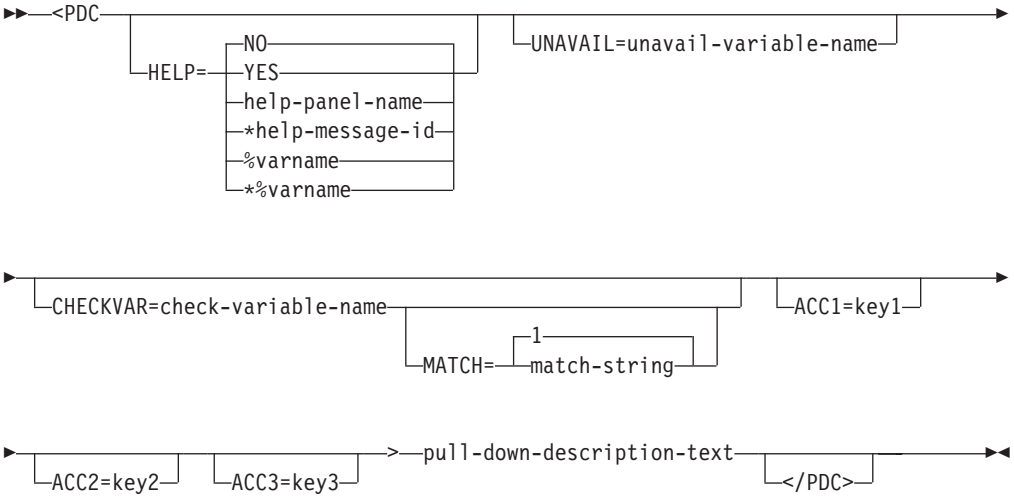
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=pd DEPTH=20>Help for Ordering Parts
<AREA>
<INFO>
  <P>Use one of the following codes when ordering
  a part number from inventory:
  <PARML TSIZE=5>
    <PT>ST
    <PD>Indicates that the part
    order is for stock replenishment.
    <PT>CU
    <PD>Indicates that the part
    order is for immediate customer shipment.
    <PT>EL
    <PD>Indicates that the part
    order is for shipment to an external location.
  </PARML>
</INFO>
</AREA>
</HELP>
```

```
            Help for Ordering Parts

Use one of the following codes when ordering a
part number from inventory:

ST
     Indicates that the part order is for stock
     replenishment.

CU
     Indicates that the part order is for
     immediate customer shipment.

EL
     Indicates that the part order is for
     shipment to an external location.

 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 151. Parameter Descriptions

PDC (Pull-Down Choice)

The PDC tag defines a pull-down choice for an action bar pull-down.



HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname

This attribute specifies the help action taken when the user requests help for a pull-down choice selection.

When HELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help on a choice and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

Note: This attribute is valid only when the SELFLD tag has been specified with TYPE=MULTI.

UNAVAIL=unavail-variable-name
This attribute specifies the name of a variable that is used by ISPF to determine the availability of the pull-down choice. When the variable value is 1, the pull-down choice is unavailable.

The *unavail-variable-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

CHECKVAR=check-variable-name
This attribute specifies a variable whose value indicates whether or not the pull-down choice is preselected when the pull-down is displayed. If the value of the variable is equivalent to the *match-string* you specify with the MATCH attribute, the pull-down choice appears preselected. Otherwise, it does not. The *check-variable-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

Note: Unlike selection fields, ISPF does not reset the *check-variable-name* to indicate the pull-down choice the user selects. Therefore, you should code the SETVAR attribute in an ACTION tag associated with the pull-down choices when the application needs to know which pull-down choice was selected.

MATCH=1 | match-string
This attribute defines the value that causes the pull-down choice to be preselected. The value of variable specified by the CHECKVAR attribute is compared to the *match-string* value, and if they are equal, the pull-down choice appears preselected.

ACC1=key1
This attribute specifies an accelerator key to be used when operating in GUI mode. The *key1* value can be Ctrl, Shift, Alt, Insert, Delete, Backspace, F1–F12, A–Z, a–z, or 0–9.

ACC2=key2
This attribute specifies an accelerator key to be used when operating in GUI mode. The *key2* value can be be Ctrl, Shift, Alt, Insert, Delete, Backspace, F1–F12, A–Z, a–z, or 0–9.

ACC3=key3
This attribute specifies an accelerator key to be used when operating in GUI mode. The *key3* value can be Ctrl, Shift, Alt, Insert, Delete, Backspace, F1–F12, A–Z, a–z, or 0–9.

pull-down-description-text
This is the text for the pull-down choice. The maximum length of the text is 64 bytes.

Each *pull-down-description-text* is prefixed with a sequential number beginning with 1 to allow selection by number.

Description

The PDC tag defines a pull-down choice for an action bar pull-down. If you do not code any PDC tags within an ABC tag, that action bar choice will not appear on the action bar.

To provide for a pull-down selection, an input field is generated prior to the first *pull-down-description-text* that allows entry of the number of the selected pull-down choice. Since field names are being generated, the application developer should not use field names beginning with Z.

Up to three accelerator keys may be specified. ISPDTLC checks for valid combinations of ACCn attributes. Invalid combinations are reset to blank and a warning message is issued.

- Insert, Delete, Backspace, and Fn are valid single keys.
- Only one ACCn can be a function key.
- SHIFT plus A–Z, a–z, or 0–9 is not valid.
- When three keys are specified, two must be CTRL, ALT, or SHIFT.
- When two keys are specified, one must be CTRL, ALT, or SHIFT.
- No two keys can have the same value.
- The combined length of the key values including any connecting "+" characters must be 30 bytes or less.
- An accelerator key combination can be used only one time on a panel.

Conditions

- You must code the PDC tag within an ABC definition. See "ABC (Action Bar Choice)" on page 214 for a complete description of this tag.
- The maximum number of pull-down choices that will be generated is 60. However, the depth specified on the enclosing PANEL tag can further reduce this maximum number.

Nested Tags

You can code the following tag within a PDC definition:

Tag	Name	Usage	Page	Required
ACTION	Action	Multiple	217	No
COMMENT	Comment	Multiple	282	No
M	Mnemonic	Single	392	No
SOURCE	Source	Multiple	472	No

Example

The following application panel markup produces the action bar and pull-down shown in Figure 152 on page 433.

In this example, when the action bar choice **Search** is chosen, the variable *whchsrch* is tested to see if one of the pull-down choices should be preselected. If *whchsrch*=1 then the pull-down choice **Search on name** is preselected with a 1 in the pull-down selection entry field. If *whchsrch*=2 then the pull-down choice **Search on card number** is preselected with a 2 in the pull-down selection entry field. If *whchsrch* is not equal to 1 or 2, then neither pull-down choice is preselected. The

example shows the **Search on name** choice preselected. If *srch2=1,* then the UNAVAIL attribute on the pull-down choice Search on card number would cause that choice to be unavailable. The example shows the result.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 system>
  <!entity sampbody system>)>
&sampvar1;

<PANEL NAME=pdc2 KEYLIST=keylxmp>Library Card Registration
<AB>
<ABC>File
  <PDC>Add Entry
    <ACTION RUN=add>
  <PDC>Delete Entry
    <ACTION RUN=delete>
  <PDC>Update Entry
    <ACTION RUN=update>
  <PDC>Exit
    <ACTION RUN=exit>
<ABC>Search
  <PDC CHECKVAR=whchsrch MATCH=1 UNAVAIL=srch1>
            ACC1=ctrl ACC2=alt ACC3=n>Search on name
    <ACTION SETVAR=whchsrch VALUE=1>
    <ACTION RUN=search>
  <PDC CHECKVAR=whchsrch MATCH=2 UNAVAIL=srch2
            ACC1=ctrl ACC2=alt ACC3=c>Search on card number
    <ACTION SETVAR=whchsrch VALUE=2>
    <ACTION RUN=search>
<ABC>Help
  <PDC>Extended Help...
    <ACTION RUN=exhelp>
  <PDC>Keys Help...
    <ACTION RUN=keyshelp>
</AB>
&sampbody;
</PANEL>
```

```
  File  Search  Help
-------------------------------------------------------------------
       1  1.  Search on name          gistration
          *.  Search on card number

Type in client's name and case number (if applicable).

Then, select an action bar choice.

Date . . . : 11/09/89
Card No  . . ______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

Choose one of the following          Check valid branches
__ 1.  New                           _  North Branch
   2.  Renewal                       _  South Branch
   3.  Replacement                   _  East Branch
                                     _  West Branch
 Enter a command ===> ___________________________________________________
  F1=Help       F2=Split       F3=Exit       F6=KEYSHELP     F9=Swap
```

Figure 152. Pull-Down Choices

PDSEP (Pull-Down Separator)

The PDSEP tag defines a horizontal divider line on an action bar pull-down menu.

Description

The PDSEP tag defines a horizontal divider line on an action bar pull-down menu. You use the horizontal divider to separate groups of related pull-down choices.

Conditions

- The PDSEP tag can only be coded between PDC tags. All PDSEP tags found before the first PDC tag or after the last PDC tag are discarded.
- Only one PDSEP tag should be coded between PDC tags. If multiple PDSEP tags are found between PDC tags, the first one is accepted and the others are discarded.
- The PDSEP tag automatically closes an open PDC tag and all nested tags following the PDC tag.

Nested Tags

None.

Example

The following example shows how the PDSEP tag is used to draw a separator line in a pull-down menu. Figure 153 on page 435 shows the formatted result.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 sysem>
  <!entity sampbody system>)>
&sampvar1;

<PANEL NAME=pdsep KEYLIST=keylxmp>Library Card Registration
<AB>
<ABC>File
  <PDC>Add Entry
    <ACTION RUN=add>
  <PDC>Delete Entry
    <ACTION RUN=delete>
<PDC>Update Entry
  <ACTION RUN=update>
<PDSEP>
<PDC>Exit
  <ACTION RUN=exit>
BC>Search
<PDC CHECKVAR=whchsrch MATCH=1 UNAVAIL=srch1
     acc1=ctrl acc2=alt acc3=n >Search on name
  <ACTION SETVAR=whchsrch VALUE=1>
  <ACTION RUN=search>
<PDC CHECKVAR=whchsrch MATCH=2 UNAVAIL=srch2
     acc1=ctrl acc2=alt acc3=c>Search on card number
    <ACTION SETVAR=whchsrch VALUE=2>
    <ACTION RUN=search>
<ABC>Help
  <PDC>Extended Help...
    <ACTION RUN=exhelp>
  <PDC>Keys Help...
```

```
    <ACTION RUN=keyshelp>
</AB>
&sampbody;
</PANEL>
```

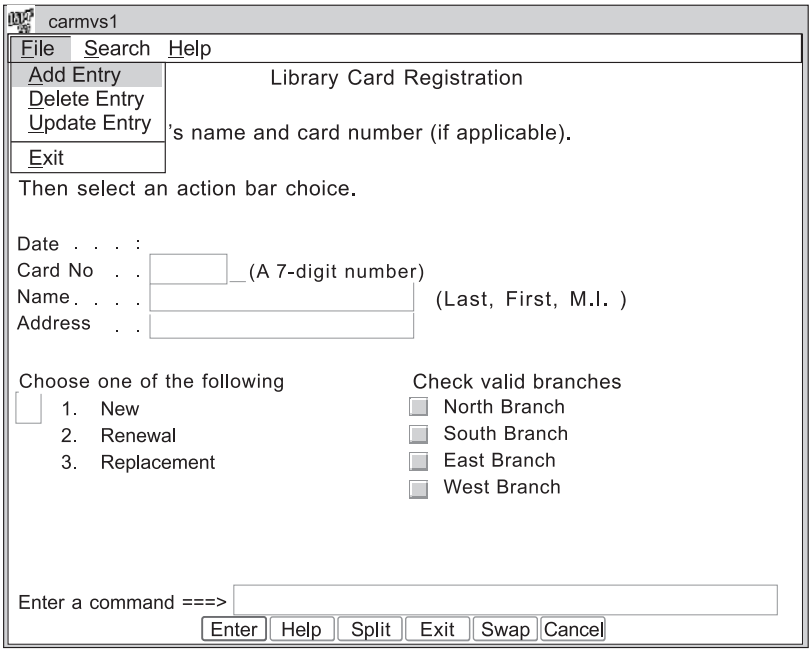
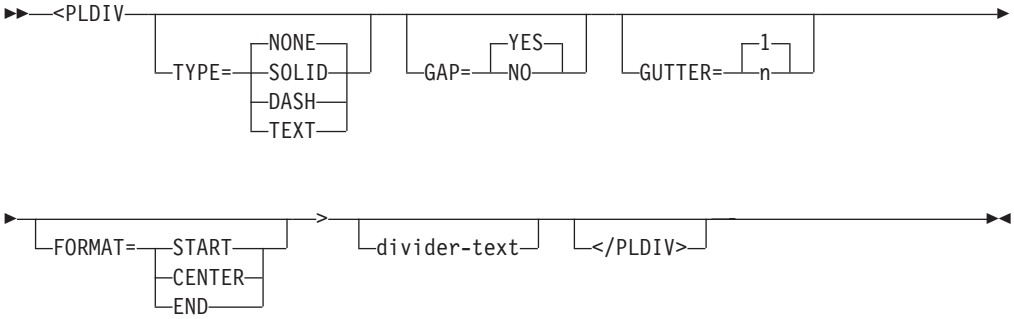


Figure 153. Pull-Down Separator

PLDIV (Parameter List Divider)

The Parameter List Divider tag greates a blank or visible divider within the text portion of a parameter list.



TYPE=NONE | SOLID | DASH | TEXT

This attribute specifies the type of parameter list divider line.

The default value is NONE, which produces a blank line. You must specify SOLID, DASH, or TEXT to produce a visible divider line. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line.

GAP=YES | NO
When GAP=NO, the divider line completely crosses from one side of the text area to the other. When GAP=YES, a one-character gap remains at each end of the divider line.

GUTTER=1 | n
This attribute specifies the total width of the parm list divider. If the GUTTER value is an even number, the conversion utility increases the number by 1 so that the divider is centered within the defined width.

The minimum GUTTER value, and the default, is 1.

FORMAT=START | CENTER | END
This attribute specifies the position of the divider text within the width of the divider line.

divider-text
This is the text of the area divider line.

Description

The PLDIV tag creates a blank or solid divider within the text portion of an application panel. A horizontally formatted visible divider is created when you specify the TYPE attribute value as SOLID or DASH. When the GRAPHIC invocation option is specified, SOLID produces a solid line for host display and DASH produces a dashed line. When NOGRAPHIC is specified or the panel is displayed in GUI mode, both SOLID and DASH produce a dashed line.

The divider line can be formatted with descriptive text. When this feature is used, the FORMAT attribute must be specified. If FORMAT is not specified, the tag text is ignored. You control the text padding with the TYPE attribute. If TYPE=TEXT, the *divider-text* is padded with blanks. When TYPE=SOLID or TYPE=DASH, the *divider-text* is padded with the specified character.

Conditions

- You must code the PLDIV tag within a PARML tag definition.

Nested Tags

You can code the following tags within a PLDIV definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No

Example

The following example illustrates the use of the PLDIV tag. Figure 154 on page 437 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=pldiv DEPTH=22 WIDTH=60>Part Number Code Help
<AREA>
<INFO>
  <P>Valid part numbers consist of a three-digit
  number followed by a 2-character suffix.
  <DIVIDER>
  <PARML TSIZE=6 compact>
    <PLDIV TYPE=solid>
    <PT>123
    <PD>The first three digits represent
```

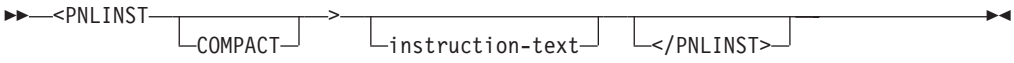
```
    the lot number of the part.
    <PLDIV TYPE=solid>
    <PT>AA
    <PD>The 2-character suffix represents the
    department the part originated from.
    The valid suffixes are:
      <PARML BREAK=none COMPACT SKIP>
        <PT>TO
        <PD>Tools
        <PT>EL
        <PD>Electrical
        <PT>ME
        <PD>Mechanical
      </PARML>
  </PARML>
</INFO>
</AREA>
</HELP>
```

```
                  Part Number Code Help

Valid part numbers consist of a three-digit number
followed by a 2-character suffix.

------------------------------------------------------------
123
      The first three digits represent the lot number of
      the part.
------------------------------------------------------------
AA
      The 2-character suffix represents the department the
      part originated from. The valid suffixes are:

      TO        Tools
      EL        Electrical
      ME        Mechanical
```

Figure 154. Parameter List Divider

PNLINST (Panel Instruction)

The PNLINST tag defines panel instructions for an application panel.



COMPACT
This attribute causes the panel instruction to format without a blank line before the text.

instruction-text
This is the text of the panel instruction. The *instruction-text* must fit in the remaining panel depth.

Description

The PNLINST tag defines panel instructions for an application panel. The *instruction-text* formats as a paragraph based on the width of the application panel, area, or region. You can code multiple paragraphs of instruction text by using a new panel instruction tag for each new paragraph.

If the COMPACT attribute is not specified, the conversion utility inserts a blank line before the panel instruction text.

Conditions

- You must code the PNLINST within a PANEL, AREA, or REGION definition.

Nested Tags

You can code the following tags within a PNLINST definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

The following application panel markup contains one panel instruction. Figure 155 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=selcls TYPE='char 2'>
<VARLIST>
   <VARDCL NAME=loc VARCLASS=selcls>
   <VARDCL NAME=mode VARCLASS=selcls>
</VARLIST>

<PANEL NAME=pnlinst HELP=trvlhlp WIDTH=60 DEPTH=22 KEYLIST=keylxmp>
Dream Vacation Guide
<AB>
  <ABC>File
    <PDC>Add Entry
             <ACTION RUN=add>
     <PDC>Delete Entry
             <ACTION RUN=delete>
     <PDC>Update Entry
             <ACTION RUN=update>
     <PDC>Exit
             <ACTION RUN=exit>
   <ABC>Help
      <PDC>Extended Help...
             <ACTION RUN=exhelp>
      <PDC>Keys Help...
             <ACTION RUN=keyshelp>
</AB>
<AREA>
  <PNLINST>Choose one of the following exotic locations and
  your preferred mode of travel, then press Enter.
  <DIVIDER>
  <REGION DIR=horiz>
  <SELFLD NAME=loc PMTWIDTH=23 SELWIDTH=25>Exotic Location:
        <CHOICE>Athens, GA
        <CHOICE>Berlin, CT
        <CHOICE>Cairo, IL
```

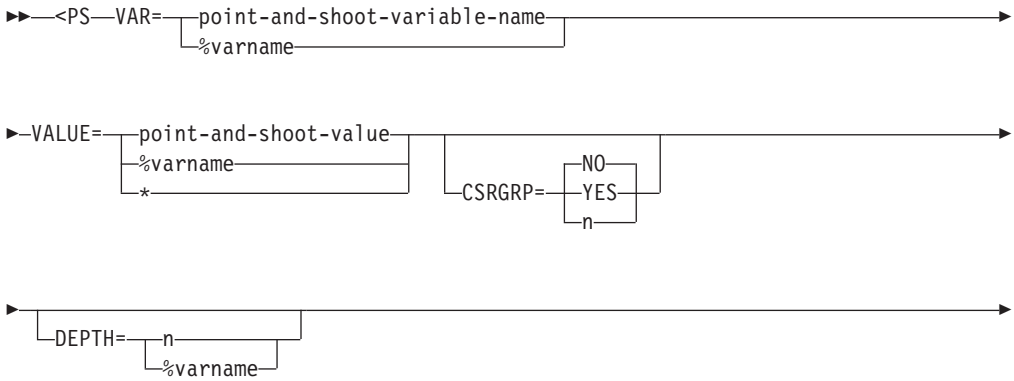
```
        <CHOICE>Lizard Lick, NC
        <CHOICE>Paris, TX
        <CHOICE>Rome, NY
        <CHOICE>Venice, FL
  </SELFLD>
  <DIVIDER>
  <SELFLD NAME=mode PMTWIDTH=25 SELWIDTH=25>Travel Mode:
        <CHOICE>Boxcar
        <CHOICE>Hitchhike
        <CHOICE>Mule
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>
</PANEL>
```

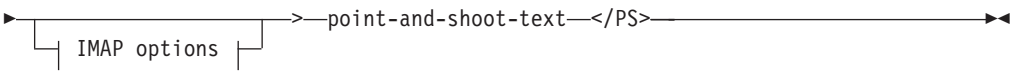
```
                    Dream Vacation Guide
Command ===> _____________________________________________

Choose one of the following exotic locations and your
preferred mode of travel, then press Enter.

Exotic Location:            Travel Mode:
__  1.  Athens, GA          __  1.  Boxcar
    2.  Berlin, CT              2.  Hitchhike
    3.  Cairo, IL               3.  Mule
    4.  Lizard Lick, NC
    5.  Paris, TX
    6.  Rome, NY
    7.  Venice, FL
```

Figure 155. Panel Instructions

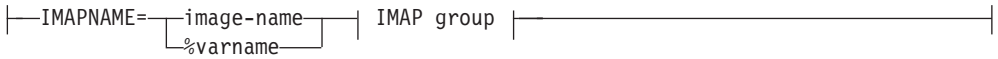
PS (Point-and-Shoot)

The PS tag defines a text string that is to be enabled for point-and-shoot.

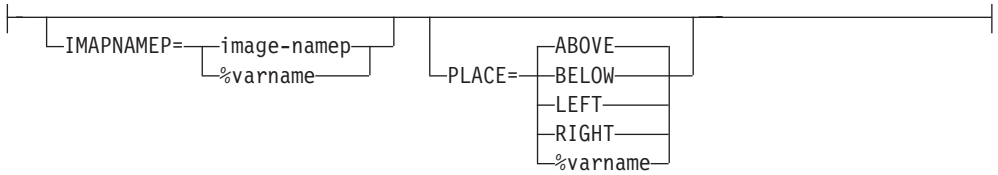




IMAP options



IMAP group



VAR=point-and-shoot-variable-name | %varname
This attribute provides the name of a variable which is to be set when a point-and-shoot phrase is clicked on for selection. You can define this attribute as a variable name preceded by a "%".

The *point-and-shoot-variable-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

VALUE=point-and-shoot-value | %varname | *

This attribute provides the value to be placed in the field specified by the VAR attribute. You can define this attribute as a variable name preceded by a "%". To specify a blank value, the "' '" (quotation mark, apostrophe, blank, apostrophe, quotation mark) coding notation should be used.

When the PS tag is used with the CHOICE tag, VALUE=* can be used to automatically use the current choice number (or SELCHAR value) as the point-and-shoot selection value.

CSRGRP=NO | YES | n
When CSRGRP=YES, the conversion utility generates a cursor group number to be used for this point-and-shoot text field. When CSRGRP=n, the number provided is used for this field.

DEPTH=n | %varname
This attribute defines the depth reserved for the point-and-shoot field. When the panel is displayed in GUI mode, the resulting push button is displayed with the specified DEPTH. You use this attribute in combination with the IMAPNAME attribute to provide space for the image. The minimum value is 1 and the maximum value is the remaining panel depth.

IMAPNAME=image-name | %varname
This attribute specifies the name of a image to be placed on the point-and-shoot push button when it is displayed in GUI mode. The *image-name* is not used when the panel is displayed in host mode.

The *image-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

IMAPNAMEP=image-namep | %varname
This attribute specifies the name of a image to be placed on the point-and-shoot push button after it has been pushed when it is displayed in GUI mode. The *image-namep* is not used when the panel is displayed in host mode.

The *image-namep* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

PLACE=ABOVE | BELOW | LEFT | RIGHT | %varname
This attribute specifies the position of the image relative to the text within the point-and-shoot push button.

point-and-shoot-text
This is the text of a point-and-shoot entry.

Description

The PS tag is valid as part of the text following:

INFO TAGS
ATTENTION, CAUTION, DD, DDHD, DT, DTHD, FIG, FIGCAP, H2, H3, H4, LI, LINES, LP, NOTE, NT, P, PD, PT, WARNING, and XMP.

PANEL TAGS
BOTINST, CHOFLD, CHOICE, DTAFLD, DTAFLDD, GRPHDR, LSTCOL, LSTGRP, PNLINST, SELFLD, and TOPINST.

The *point-and-shoot-text* is color emphasized within the text of the panel. When running in GUI mode, the *point-and-shoot-text* displays as a push button. For host displays, the user places the cursor on the *point-and-shoot-text* and presses ENTER to select the option.

Conditions

- The PS tag requires an end tag.
- When a "%varname" notation is found on any of the attributes that allow a variable name, the "%varname" entry must follow the standard naming convention described in "Rules for "%variable" Names" on page 211.

Nested tags

None.

Example

The following example is provided to show the use of point-and-shoot selection for a sample option menu. Figure 156 on page 442 shows the formatted result.

```
<!doctype dm system ()>
<!-- Sample selection menu with point-and-shoot -->
<panel name=ps1 menu keylist=keylxmp>Sample Point-and-Shoot
  <topinst>This is a selection panel.
  <selfld type=menu   pmtloc=before
          selwidth=40 pmtwidth=10>Select an option
    <choice checkvar=xtest1 match=a>
        <PS VAR=zcmd VALUE=1>Selection #1 (Command Tstch1)</PS>
      <action run=tstch1 parm='1 2 3 4'
       passlib newpool suspend>
    <choice checkvar=xtest1 match=b>
        <PS VAR=zcmd VALUE=2>Selection #2 (Command Tstch2)</PS>
      <action run=tstch2 parm=1234>
    <choice checkvar=xtest1 match=c>
```

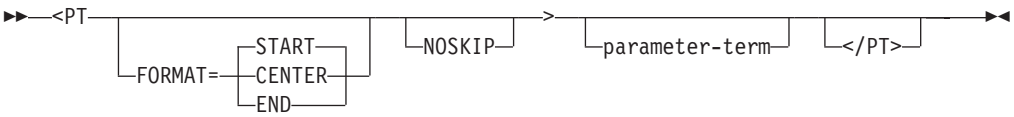
```
        <PS VAR=zcmd VALUE=3>Selection #3 (Command Tstch3)</PS>
      <action run=tstch3 parm=abcd>
    <choice checkvar=xtest1 match=d>
        <PS VAR=zcmd VALUE=4>Selection #4 (Command Tstch4)</PS>
      <action run=tstch4 parm='a b c d'>
  </selfld>
  <cmdarea>
</panel>
```

```
                        Sample Point-and-Shoot

 This is a selection panel.

 Select an
 option . . 1  Selection #1 (Command Tstch1)
            2  Selection #2 (Command Tstch2)
            3  Selection #3 (Command Tstch3)
            4  Selection #4 (Command Tstch4)









 Option ===> _____________________________________________________________
  F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
 F12=Cancel
```

Figure 156. Point-and-Shoot Fields

PT (Parameter Term)

The PT tag defines a term in a parameter list.



FORMAT = START | CENTER | END
This attribute specifies the placement of the PT tag text within the space provided by TSIZE. The PT tag FORMAT setting overrides the FORMAT setting of the enclosing PARML tag.

NOSKIP
This attribute causes the definition term to be formatted without a blank line before the term. It is used to control the formatting of the parameter term when COMPACT has not been specified on the enclosing PARML tag. When the PARML tag TSIZE attribute specifies that multiple PT tags are to be formatted for each PD tag, NOSKIP should be coded on the first PT tag. It is ignored for the second and subsequent PT tags.

parameter-term
This is the text of the parameter term.

Description

The PT tag defines a parameter term in a parameter list.

Conditions

- You must code the PT tag within a PARML definition. See "PARML (Parameter List)" on page 425 for a complete description of this tag.
- Each PT tag must be paired with an associated PD tag. You can specify only one PT tag for each PD tag within a parameter list. The PT tag must immediately precede the PD tag it is associated with.

Nested Tags

You can code the following tag within a PT definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

The following help panel markup contains a parameter list with two parameter terms. Figure 157 on page 444 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=pt WIDTH=40 DEPTH=18>Help for the Duplex Function
<AREA>
<INFO>
  <P>The two options associated with
  the DUPLEX function are:
  <PARML TSIZE=5>
    <PT>DCopies
    <PD>Which prints one-sided copies that
    are prepared for future duplex copying.
    <PT>DPrint
    <PD>Which prints two-sided copies.
  </PARML>
</INFO>
</AREA>
</HELP>
```

```
     Help for the Duplex Function

The two options associated with the
DUPLEX function are:

DCopies
     Which prints one_sided copies
     that are prepared for future
     duplex copying.

DPrint
     Which prints two_sided copies.



 F1=Help      F3=Exit      F5=Exhelp
 F6=Keyshelp  F7=PrvTopic  F8=NxtTopic
F10=PrvPage  F11=NxtPage  F12=Cancel
```

Figure 157. Parameter Terms

PTDIV (Parameter Term Divider)

The PTDIV tag defines a visible vertical divider (|) between multible PT tags.

Description

The PTDIV tag can be used to create a visual separation between the parameter terms. Each PTDIV tag adds a vertical bar (plus display control attributes) to the parameter list..

Conditions

The PTDIV tag can be coded before the first PT tag, between PT tags, or following the last PT tag (before the PD tag definition).

Nested Tags

None.

Example

The following example illustrates the use of the PTDIV tag in combination with the DIVEND attribute of the PARML tag. Figure 158 on page 445 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=ptdiv DEPTH=22 WIDTH=60>Part Number Code Help
<AREA>
<INFO>
  <P>Valid part numbers consist of a three-digit
  number followed by a 2-character suffix.
  <DIVIDER>
  <PARML TSIZE=6 compact>
    <PLDIV TYPE=solid>
```

```
    <PT>123
    <PD>The first three digits represent
    the lot number of the part.
    <PLDIV TYPE=solid>
    <PT>AA
    <PD>The 2-character suffix represents the
    department the part originated from.
    The valid suffixes are:
      <PARML BREAK=none COMPACT SKIP DIVEND=yes>
        <PLDIV TYPE=solid>
        <PTDIV>
        <PT>TO
        <PTDIV>
        <PD>Tools
        <PTDIV>
        <PT>EL
        <PTDIV>
        <PD>Electrical
        <PTDIV>
        <PT>ME
        <PTDIV>
        <PD>Mechanical
      </PARML>
  </PARML>
</INFO>
</AREA>
</HELP>
```

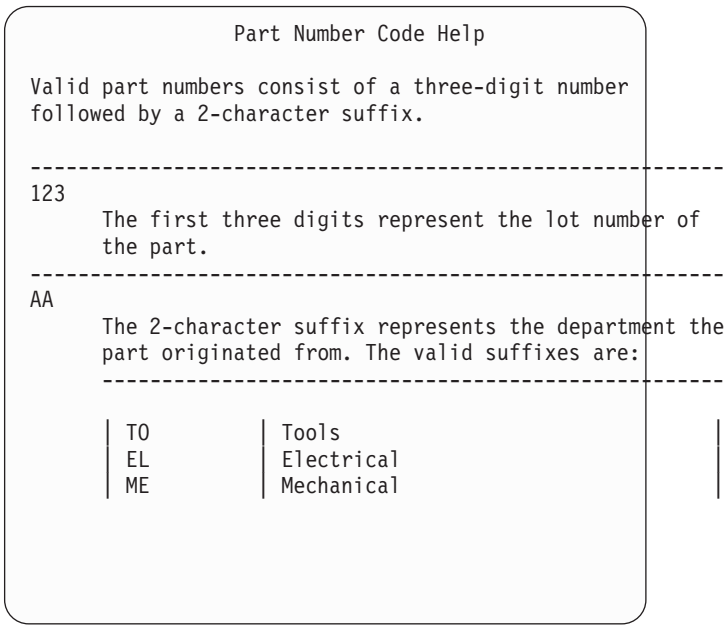
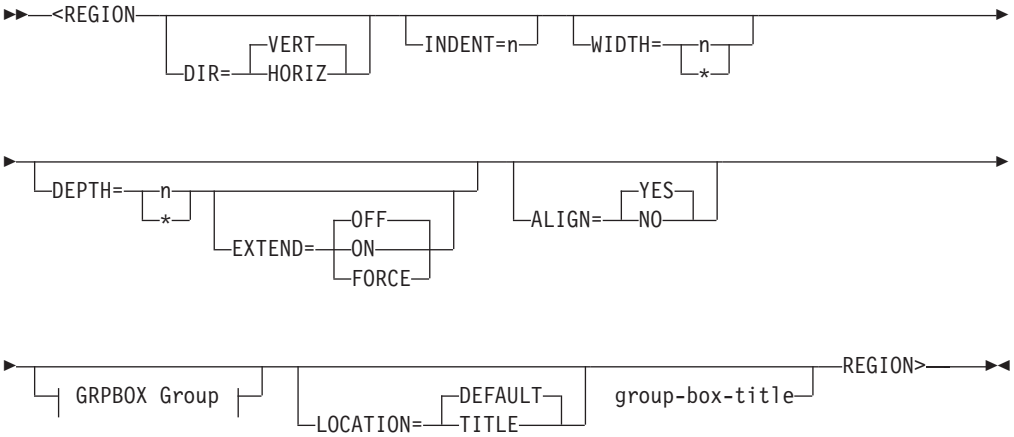


Figure 158. Parameter Term Divider

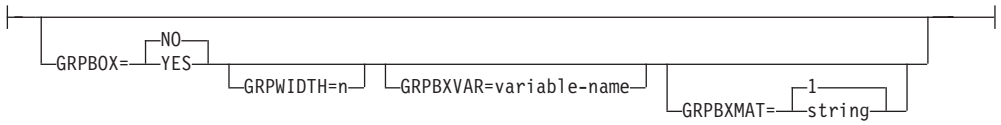
REGION (Region)

The REGION tag defines the characteristics of a panel section including the direction in which fields on an application panel are arranged.

REGION



GRPBOX Group:



DIR=VERT | HORIZ
This attribute specifies in which direction the contents of a region is arranged. The default value is VERT, which formats the contents of the region in a vertical direction; that is, top to bottom. If you specify the HORIZ value for DIR, the contents of the region are formatted horizontally; that is, left to right within the region.

INDENT=n
This attribute defines the number of columns to indent the current region from the current left region boundary.

WIDTH=n | *
This attribute defines the width of a panel region. If WIDTH is not specified or WIDTH=*, the default value is the remaining available panel width.

DEPTH=n | *
This attribute defines the size of a scrollable region. When EXTEND=OFF, the minimum value is 2 and the maximum value is the remaining panel depth. When EXTEND=ON, the minimum value is 1. If the DEPTH value is specified as "*", the conversion utility will reserve the remaining available panel depth for the scrollable region.

If DEPTH is not specified the region will not be scrollable.

EXTEND=OFF | ON | FORCE
This attribute defines the run-time display size for the scrollable region. If EXTEND=ON is specified, the panel definition is expanded from the minimum DEPTH to the size of the logical screen. Only one EXTEND=ON attribute value is allowed on a panel. The first tag (AREA, DA, GA, REGION, SELFLD) with EXTEND=ON is accepted; the EXTEND attribute on any subsequent tag is ignored.

If you intend to display the panels in a pop-up window, it is recommended that you code EXTEND=OFF.

If the EXTEND attribute is specified without the DEPTH attribute, a warning message is issued and the EXTEND attribute is ignored.

If EXTEND=FORCE is specified within a horizontal area or region, the EXTEND(ON) keyword is added to the scrollable area attribute statement in the)ATTR panel section. The conversion utility issues a message to advise of a potential display error if other panel fields are formatted on or after the last defined line of the scrollable area.

ALIGN=YES | NO
This attribute controls the horizontal alignment of the first fields in horizontal regions. The default is to align the fields to facilitate cursor movement by tabbing. This attribute is valid only when DIR=HORIZ.

GRPBOX=NO | YES
This attribute is used to specify a group box. The default value is NO. The group box outline is visible only when running ISPF in GUI mode.

When GRPBOX=YES is specified on the same REGION tag that defines a scrollable region, the group box title is formatted as the first line within the)AREA panel section.

GRPWIDTH=n
This attribute is used to specify the width of the group box. The default and maximum group box width is the region width.

GRPWIDTH can be used to specify a group box width smaller than the default value. As an example, when the region consists only of a SELFLD tag that is formatted into multiple columns for host display, but is specified as a list box or drop-down list, the GUI mode display appears as a single column. The right border of the group box would normally extend beyond the space required for the GUI display. The GRPWIDTH attribute can be used to limit the group box to the width of the list box or drop-down list.

GRPBXVAR=variable-name
This attribute defines a variable whose value indicates whether or not the group box outline is added when the panel is displayed in GUI mode. If the variable is equal to the value specified by the GRPBXMAT attribute, the group box outline is added.

The GRPBXVAR attribute value must be specified without a leading % sign. The *variable-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

GRPBXMAT=1 | string
This attribute defines the value for the GRPBXVAR variable that indicates the group box outline is to be added to the panel in GUI mode. The *string* can be any character string. GRPBXMAT=1 is the default.

LOCATION=DEFAULT | TITLE
This attribute is used to build a panel 'title' which requires data fields in specific column positions. A single line may be formatted to be placed in the panel title position by enclosing the appropriate tags within a horizontal region specifying LOCATION=TITLE. The resulting line displays with the colors associated with the tags used to format the line. This attribute is valid only when DIR=HORIZ.

group-box-title
This is the title for the group box. The *group-box-title* should be supplied only when GRPBOX=YES. In other cases a warning message will be issued.

Description

The REGION tag defines the characteristics of a panel section. You can code multiple regions within an application panel.

Nonscrollable horizontal regions are normally aligned left-to-right using the first input field from each region. If a panel consists of both scrollable and nonscrollable regions formatted horizontally, scrollable regions will normally be aligned with the first input fields of nonscrollable regions.

Regions containing data formatted from INFO tags or from the GRPHDR tag will normally start with a blank line when formatted in the)BODY panel section. The blank line is omitted when these tags are formatted at the beginning of a scrollable area.

If you specify the CMDAREA tag within your DTL source file, it must appear before the REGION tag when DEPTH=* is specified. The REGION tag DEPTH may have to be adjusted to allow for additional lines which result from tags present within the panel definition following the end REGION tag.

Conditions

- The REGION tag requires an end tag.
- You must code the REGION tag within an AREA or PANEL definition. See "AREA (Area)" on page 223 and "PANEL (Panel)" on page 415 for descriptions of these tags.
- You can also nest regions within other regions.
- You can code only one LSTFLD tag within a REGION definition.

Compatibility Considerations

In Version 3, the conversion utility did not format nested horizontal regions. In this release, nested horizontal formatting is supported.

Nested Tags

You can code the following tags within a REGION definition:

Tag	Name	Usage	Page	Required
COMMENT	Comment	Multiple	282	No
DA	Dynamic area	Multiple	287	No
DIVIDER	Area divider	Multiple	296	No
DTACOL	Data column	Multiple	306	No
DTAFLD	Data field	Multiple	312	No
GA	Graphic area	Single	333	No
GENERATE	Generate	Multiple	335	No
GRPHDR	Group header	Multiple	337	No
INFO	Information region	Multiple	356	No
LSTFLD	List field	Single	382	No
PNLINST	Panel Instruction	Multiple	437	No
REGION	Region	Multiple	382	No

Tag	Name	Usage	Page	Required
SELFLD	Selection field	Multiple	455	No

Help Panel

You can code the following tags within a REGION definition on a help panel:

Tag	Name	Usage	Page	Required
DIVIDER	Area divider	Multiple	296	No
INFO	Information region	Multiple	356	No
REGION	Region	Multiple	445	No

Example

The following application panel markup contains both horizontal and vertical regions. The first two horizontal regions arrange the fields coded within them in a horizontal format. The third horizontal region arranges the selection field and the contents of the vertical region nested within it in a horizontal format. In this example, the INDENT attribute has been used to indent all fields formatted within a region 2 positions under the previous text. The ALIGN attribute has adjusted the default placement of fields in the last vertical region. Figure 159 on page 450 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=chr25 TYPE='char 25'>
<VARCLASS NAME=chr12 TYPE='char 12'>
<VARCLASS NAME=chr10 TYPE='char 10'>
<VARCLASS NAME=chr9  TYPE='char 9'>
<VARCLASS NAME=chr8  TYPE='char 8'>
<VARCLASS NAME=chr2  TYPE='char 2'>

<VARLIST>
  <VARDCL NAME=name   VARCLASS=chr25>
  <VARDCL NAME=date   VARCLASS=chr8>
  <VARDCL NAME=addr   VARCLASS=chr25>
  <VARDCL NAME=city   VARCLASS=chr10>
  <VARDCL NAME=state  VARCLASS=chr9>
  <VARDCL NAME=zip    VARCLASS=chr12>
  <VARDCL NAME=level  VARCLASS=chr2>
  <VARDCL NAME=graddate VARCLASS=chr2>
  <VARDCL NAME=major  VARCLASS=chr10>
</VARLIST>

<PANEL NAME=region1 keylist=keylxmp>Application Form
<TOPINST>Complete all of the fields below, then press Enter.
<AREA>
  <REGION INDENT=2>
    <REGION DIR=horiz>
    <DTACOL PMTWIDTH=10>
      <DTAFLD DATAVAR=name ENTWIDTH=25>Name
      <DTAFLD DATAVAR=date ENTWIDTH=8 DESWIDTH=10>Date
        <DTAFLDD>(mm/dd/yy)
    </DTACOL>
    </REGION>
      <DTAFLD DATAVAR=addr ENTWIDTH=25 PMTWIDTH=10>Address
    <REGION DIR=horiz>
      <DTAFLD DATAVAR=city PMTWIDTH=10 ENTWIDTH=25>City
      <DTAFLD DATAVAR=state PMTWIDTH=9 ENTWIDTH=2>State
      <DTAFLD DATAVAR=zip PMTWIDTH=12 ENTWIDTH=5>Zip code
    </REGION>
```

```
  </REGION>
  <DIVIDER TYPE=solid GUTTER=3>
  <REGION DIR=horiz INDENT=2 ALIGN=no>
    <SELFLD NAME=level SELWIDTH=35 PMTWIDTH=25>Highest education level:
      <CHOICE>Some high school
      <CHOICE>High school graduate
      <CHOICE>Some college
      <CHOICE>College graduate
      <CHOICE>Some post-graduate work
      <CHOICE>Post-graduate degree
    </SELFLD>
    <DIVIDER TYPE=solid>
    <REGION>
      <GRPHDR FORMAT=none COMPACT STRIP>
          For applicants who are
          high school or college
          graduates:
      <REGION INDENT=2>
        <DTACOL PMTWIDTH=20>
          <DTAFLD DATAVAR=graddate ENTWIDTH=2>Year of graduation
          <DTAFLD DATAVAR=major ENTWIDTH=10>Field of study
        </DTACOL>
      </REGION>
    </REGION>
  </REGION>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

```
                          Application Form

Complete all of the fields below, then press Enter.

  Name . . . _____________________________  Date . . . ________  (mm/dd/yy)
  Address    _____________________________
  City . . . _____________________________  State . . __  Zip code . . _____

 ------------------------------------------------------------------------

  Highest education level:            |  For applicants who are
  __   1.  Some high school           |  high school or college
       2.  High school graduate       |  graduates:
       3.  Some college               |    Year of graduation   __
       4.  College graduate           |    Field of study . . . __________
       5.  Some post-graduate work    |
       6.  Post-graduate degree       |


 Enter a command ===> ____________________________________________________
  F1=Help       F3=Exit       F5=Display     F6=Keyshelp   F10=Actions
 F12=Cancel
```

Figure 159. Regions

The following example illustrates the use of the WIDTH and DEPTH attributes. The first vertical region width reserves the space required for the second vertical region, which is also scrollable. Figure 160 on page 452 shows the formatted result.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar2 sysem>
  <!entity sampabc system>)>
&sampvar2;
```

```
<PANEL NAME=region3 KEYLIST=keylxmp>File-A-Case
<AB>
&sampabc;
</AB>
<CMDAREA>Enter a command
<TOPINST COMPACT>
         Type in client's name and case number (if applicable).
<TOPINST>Then select an action bar choice.
<REGION DIR=horiz>
  <REGION WIDTH=50>
    <DTAFLD DATAVAR=caseno PMTWIDTH=12 ENTWIDTH=7 DESWIDTH=21>Case No
       <DTAFLDD>(A 7-digit number)
    <DTAFLD DATAVAR=name PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=8>Name
       <DTAFLDD>(Last, First, M.I.)
    <DTAFLD DATAVAR=address PMTWIDTH=12 ENTWIDTH=25>Address
    <DIVIDER>
    <SELFLD NAME=casesel PMTWIDTH=11 PMTLOC=before SELWIDTH=38>Choose
    one of the following
       <CHOICE CHECKVAR=case MATCH=civ>Civil
       <CHOICE CHECKVAR=case MATCH=estate>Real Estate
       <CHOICE CHECKVAR=case MATCH=environ>Environmental
    </SELFLD>
  </REGION>
  <REGION DEPTH=10>
    <SELFLD TYPE=multi PMTWIDTH=24 SELWIDTH=26>
      Check type of offense
       <CHOICE NAME=patin HELP=patin CHECKVAR=val>Patent Infringement
       <CHOICE NAME=defa HELP=defame CHECKVAR=def>Defamation
       <CHOICE NAME=cont HELP=cont CHECKVAR=con>Breach of Valid Contract
       <CHOICE NAME=priv HELP=priv CHECKVAR=pri>Invasion of Privacy
       <CHOICE NAME=incr HELP=incr CHECKVAR=icr>Interference with
                Contractual Relations
       <CHOICE NAME=disp HELP=disp CHECKVAR=dis>Improper Disposal of
                Medical By-Products
       <CHOICE NAME=fraud HELP=fraud CHECKVAR=fra>Fraud
    </SELFLD>
  </REGION>
</REGION>
</PANEL>
```

```
  File  Search  Help
------------------------------------------------------------------------------
                              File-A-Case

Type in client's name and case number (if applicable).
Then select an action bar choice.
                                                 #SAREA37                  #
Case No  . . _______  (A 7-digit number)         #                         #
Name . . . . _____________________________  (Last,   #                     #
                                           First,   #                     #
                                           M.I.)    #                     #
Address  . . _____________________________          #                     #
                                                    #                     #
Choose one                                          #                     #
of the                                              #                     #
following   __  1.  Civil                           #                     #
                2.  Real Estate
                3.  Environmental

Enter a command ===> ____________________________________________________
 F1=Help       F3=Exit       F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

The contents of the scrollable area are as follows:

```
    )AREA SAREA37
```

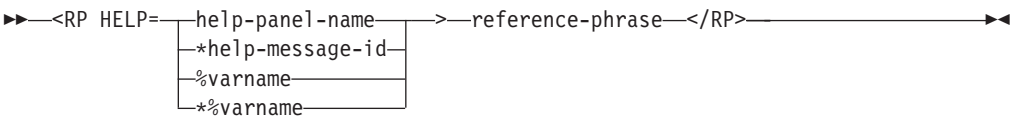
```
Check type of offense
_  Patent Infringement
_  Defamation
_  Breach of Valid
   Contract
_  Invasion of Privacy
_  Interference with
   Contractual Relations
_  Improper Disposal of
   Medical By-Products
_  Fraud
```

```
    )AREA SAREA37
```

Figure 160. Using WIDTH and DEPTH Attributes

RP (Reference Phrase)

The RP tag specifies a word or phrase within panel text that has additional help information associated with it.



HELP= help-panel-name | *help-message-id | %varname | *%varname
This attribute specifies the name of a panel that displays when the user requests help for the *reference-phrase*.

You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help on a choice and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

reference-phrase
This is the text of the phrase.

Description

The RP tag specifies a word or phrase within panel text that has additional information associated with it.

The RP tag is valid as part of the text following:

INFO tags
ATTENTION, CAUTION, DD, DDHD, DT, DTHD, FIG, FIGCAP, H2, H3, H4, LI, LINES, LP, NOTE, NT, P, PD, PT, WARNING, and XMP.

PANEL tags
BOTINST, CHOFLD, CHOICE, DTAFLD, DTAFLDD, GRPHDR, LSTCOL, LSTGRP, PNLINST, SELFLD, and TOPINST.

The *reference-phrase* is emphasized within the text of the panel to inform the user that additional information is available. The user positions the cursor on the reference phrase and presses F1=Help to obtain help on the phrase.

Each reference phrase is related to additional help panels in a manner similar to field-level help. The panel that appears when you request help from a reference phrase can also contain reference phrases.

Each *reference-phrase* results in one or more entries in the)HELP panel section. Multiple entries are required for phrases that span lines; a separate entry is created for each panel line used by the *reference-phrase*.

Conditions

- The RP tag requires an end tag.

Nested Tags

None.

Example

The following help panel markup contains a reference phrase definition for the phrase, “lifetime warranty”. Figure 161 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=rp>HELP for Appliances
<AREA>
<INFO>
<p>In addition to our free delivery and installation program, we also
offer an exclusive <rp help=warrtyh>lifetime warranty</rp> on all
of our appliances.
</INFO>
</AREA>
</HELP>

<help name=warrtyh>Help for Lifetime Warranty
<AREA>
<INFO>
<p>Lifetime warranty covers the replacement of any part that breaks
or becomes non-functional while this product is used by the original
owner.
</INFO>
</AREA>
</HELP>
```

```
                Help for Appliances

In addition to our free delivery and
installation program, we also offer an exclusive
lifetime warranty on all of our appliances.


 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 161. Reference Phrase Example

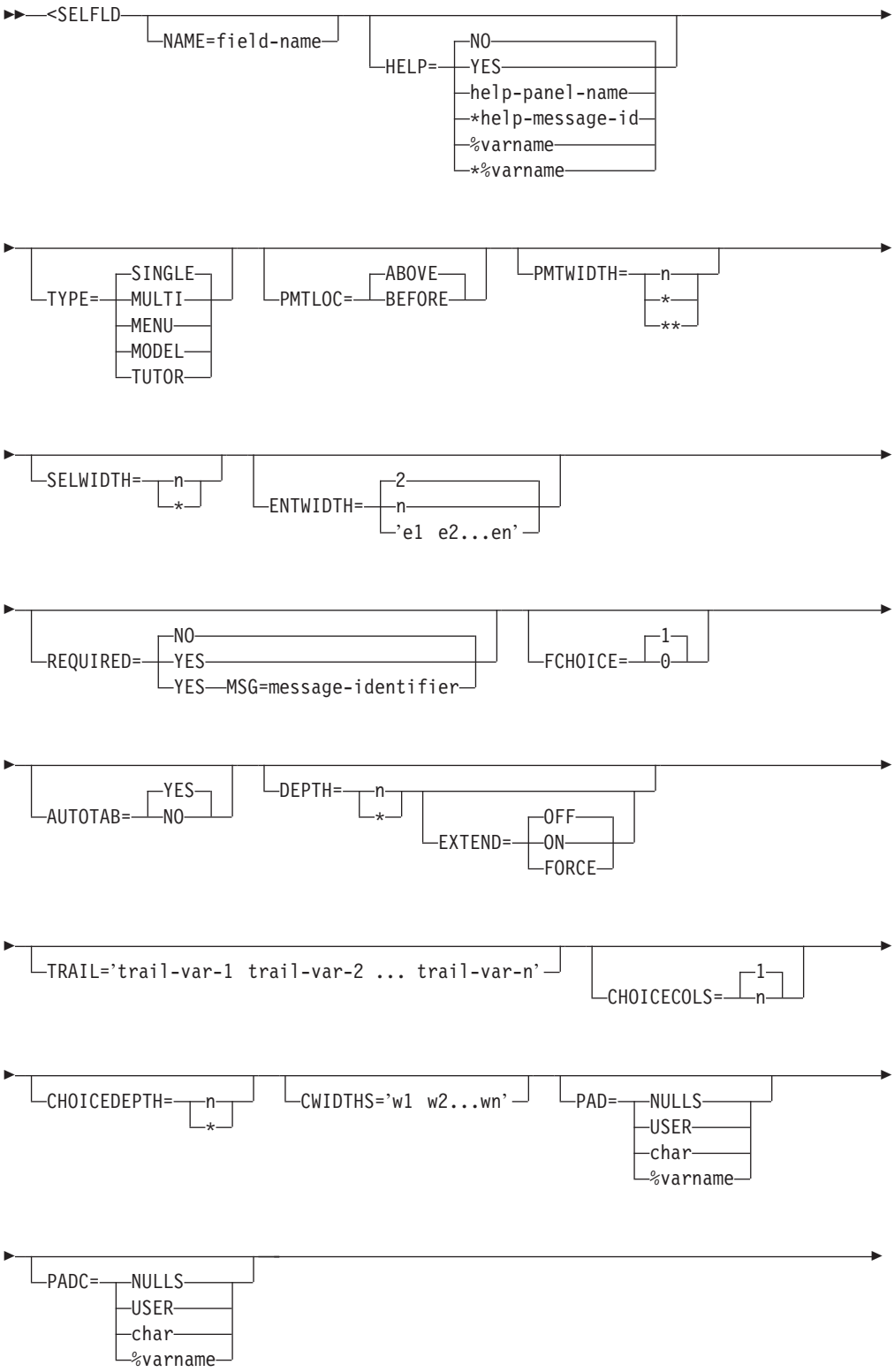
Accordingly, when the user selects the reference phrase **lifetime warranty**, the help panel specified by the HELP attribute (help=warrtyh) is displayed. Figure 162 shows the formatted result.

```
           Help for Lifetime Warranty

Lifetime warranty covers the replacement of any
part that breaks or becomes non-functional while
this product is used by the original owner.


 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

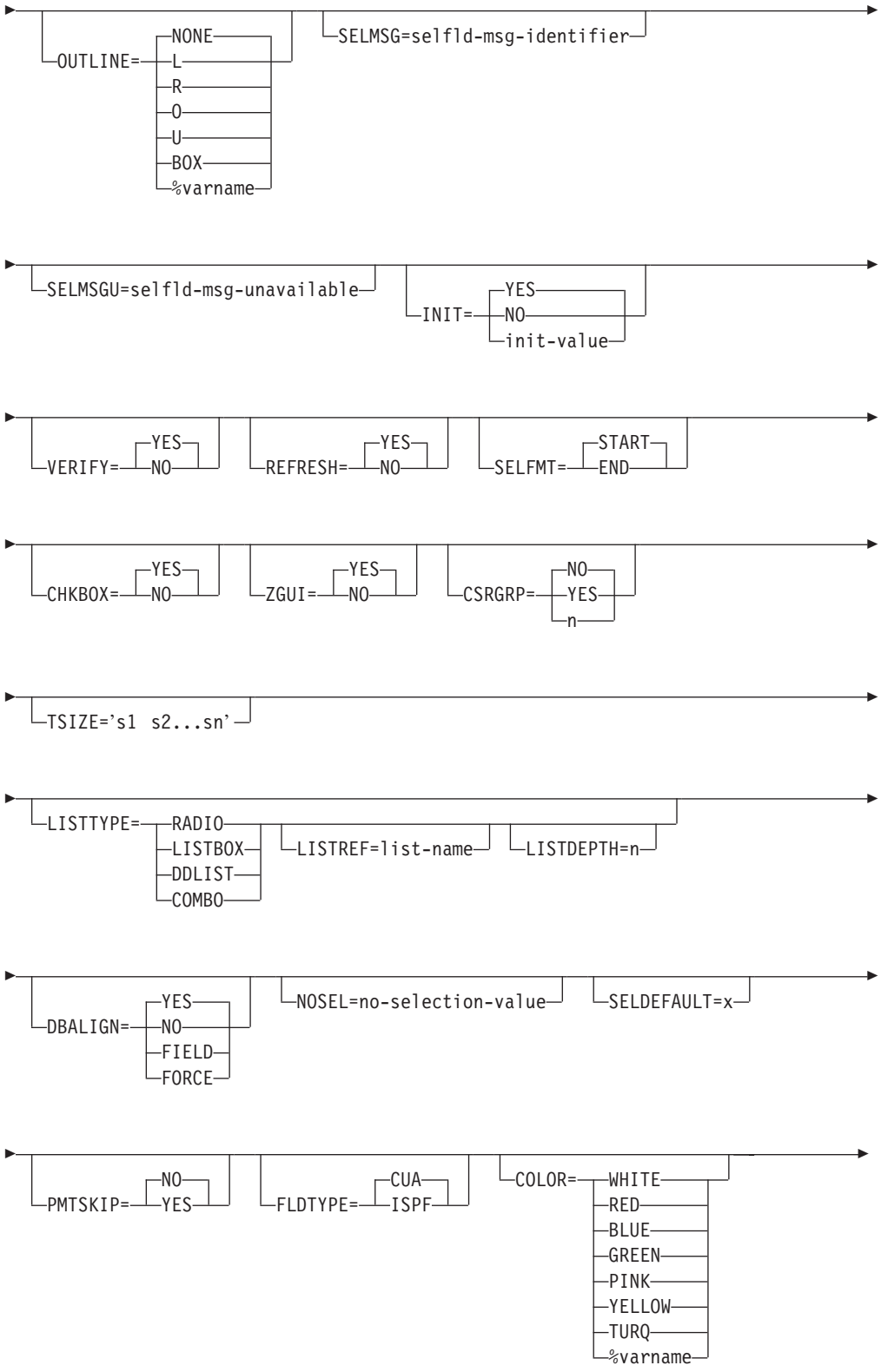
Figure 162. Reference Phrase Example

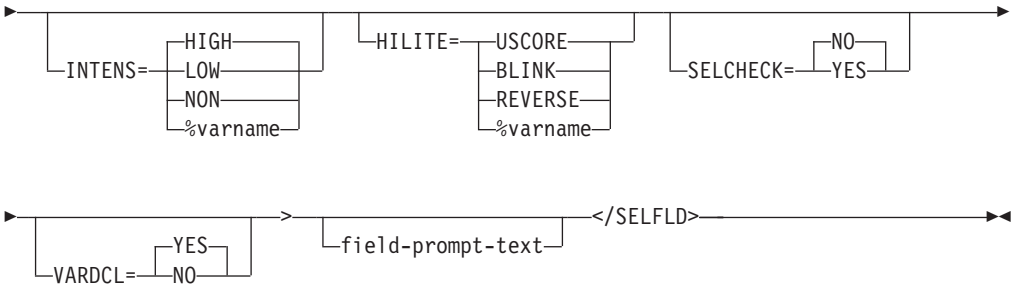
SELFLD (Selection Field)

The SELFLD tag defines a field that includes a list of choices.



SELFLD


OUTLINE=
NONE
L
R
O
U
BOX
%varname
SELMSG=selfld-msg-identifier
SELMSGU=selfld-msg-unavailable
INIT=
YES
NO
init-value
VERIFY=
YES
NO
REFRESH=
YES
NO
SELFMT=
START
END
CHKBOX=
YES
NO
ZGUI=
YES
NO
CSRGRP=
NO
YES
n
TSIZE='s1 s2...sn'
LISTTYPE=
RADIO
LISTBOX
DDLIST
COMBO
LISTREF=list-name
LISTDEPTH=n
DBALIGN=
YES
NO
FIELD
FORCE
NOSEL=no-selection-value
SELDEFAULT=x
PMTSKIP=
NO
YES
FLDTYPE=
CUA
ISPF
COLOR=
WHITE
RED
BLUE
GREEN
PINK
YELLOW
TURQ
%varname



NAME=field-name
This attribute specifies the name for the selection field. The *field-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

The NAME field is required if TYPE=SINGLE because the selection field name is used as the input field for single-choice selection fields. The NAME field is ignored if TYPE=MULTI.

The NAME field is optional for TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR. If present, it is used in place of the command field name in the construction of the option selection statement. However, because the input field is the command line, you must provide panel logic using the SOURCE tag to ensure that the selection choice is placed in the NAME field.

For single-choice selection fields, the *field-name* can be used to position the cursor on the field using the CURSOR attribute of the enclosing PANEL tag or the CURSOR parameter of the DISPLAY service call. In addition, you can use the *field-name* to position a pop-up using the POPLOC parameter of the ADDPOP service call.

HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname
This attribute specifies the help action taken when the user requests help for a selection field. This is field-level help.

When HELP=YES, control is returned to the application. You can specify either a help panel or a message identifier. If a message identifier is used, it must be prefixed with an asterisk (*).

The help attribute value can be specified as a variable name. When **%varname** is coded, a panel variable name is created. When ***%varname** is coded, a message variable name is created.

If the user requests help on a field and no help is defined, the extended help panel is displayed. If an extended help panel is not defined for the panel, the application or ISPF tutorial is invoked.

The *help-panel-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

See "HELP (Help Panel)" on page 340 for information on creating help panels. For information about creating messages, see "MSG (Message)" on page 393.

Note: This attribute is valid only when TYPE=SINGLE.

TYPE=SINGLE | MULTI | MENU | MODEL | TUTOR
This attribute specifies whether the selection field is single-choice, multiple-choice, an ISPF selection menu, an edit model selection menu, or a tutorial selection menu.

Single-choice selection fields allow the user to select only one choice from the selection list. Choices in a single-choice selection field appear with sequential numbers before each choice. An input field precedes the first choice in the selection field.

Multiple-choice selection fields allow the user to select one or more choices from the selection list. Choices in a multiple-choice selection field appear with a single character input field before each choice.

The use of TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR is allowed only when the MENU keyword has been specified on the PANEL tag. ISPF selection menu, edit model, or tutorial selection menu fields are formatted in a manner similar to single-choice selection fields. Choices appear with sequential numbers before each choice and the user may select only one choice from the selection list. With these options, the command line is used as the entry choice field. Because the HELP attribute on the SELFLD tag is not valid when TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR, help for selection menu or edit model menu choices must be entered on the CMDAREA tag.

Note: Because the selection menu, edit model menu, or tutorial menu panel uses the command line for choice selection, a command area is required. The conversion utility will automatically generate a command area if no CMDAREA tag is provided.

PMTLOC=ABOVE | BEFORE
This attribute specifies whether the *field-prompt-text* appears above or in front of the selection field.

PMTWIDTH=n | * | **
This attribute specifies the number of bytes to be used by the prompt for the selection field. When you specify PMTWIDTH=*, the conversion utility uses the length of the prompt text as the prompt width. When you specify PMTWIDTH=**, the conversion utility uses the maximum available space as the prompt width. If any prompt is longer than this value, the prompt is word-wrapped to fit on multiple lines. The minimum value is 0 and the maximum is the remaining available panel (or region) value. This value overrides the PMTWIDTH value on an enclosing DTACOL tag.

SELWIDTH=n | *
This attribute specifies the number of bytes used for the choices in the selection field. It is useful for defining a consistent appearance for the selection choices. If you do not specify the SELWIDTH parameter on the SELFLD tag, the SELWIDTH parameter on any enclosing DTACOL tag is used. If you do not specify a SELWIDTH value and SELWIDTH is not specified on an enclosing DTACOL tag, then the remaining available width of the panel (or current region) determines the width used to format the choice text. If the SELWIDTH value is specified as "*", the conversion utility will use the remaining available width.

If the width required by the *choice-description-text* and its entry-field exceeds the value specified for SELWIDTH, the text is word-wrapped to multiple lines.

Note: Because all of the remaining space is used if no SELWIDTH attribute is provided or if SELWIDTH="*" is coded, you should specify a SELWIDTH value for fields defined:
- With PMTLOC=BEFORE, because PMTWIDTH is not part of the space reserved by SELWIDTH.
- Within a horizontal region if additional fields are to be formatted to the right of the SELFLD section.

SELWIDTH for selection fields defined within a horizontal region if additional fields are to be formatted to the right of the SELFLD section.

The width specified for a single-choice selection field should include all or a portion of the *choice-description-text* plus 8-13 positions, determined as follows:
- The choice selection entry-field (1-3 characters)
- The entry-field 3270 attributes (2 characters)
- The choice-number inserted by the conversion utility (3-5 characters)
- The 3270 attributes that enclose the *choice-description-text* (2 characters).

The width of a multiple-choice selection field should include all or a portion of the *choice-description-text* plus 5 positions, determined as follows:
- The choice selection entry-fields (1 character)
- The entry-field 3270 attributes (2 characters)
- The 3270 attributes that enclose the *choice-description-text* (2 characters).

The width specified for a menu-choice, model-choice, or tutorial-choice selection field should include all or a portion of the choice-description-text plus 4-19 positions, determined as follows:
- The choice selection entry-field (1-16 characters)
- The entry-field 3270 attribute (1 character)
- The 3270 attributes that enclose the choice-description-text (2 characters).

ENTWIDTH=2 | n | 'e1 e2...en'

This attribute is valid only when TYPE=SINGLE, TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR.

Multiple ENTWIDTH values can be used when TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR. For these types of selection lists, the ENTWIDTH is used only to format the amount of space used by the selection character(s). The multiple width format is used when CHOICECOLS is greater than 1 to customize the width required for each column of choices. If the number of ENTWIDTH values is less than the number of columns, the last (or only) ENTWIDTH value is used for the remaining columns. If more ENTWIDTH values are supplied than there are columns of choices, the excess ENTWIDTH values are ignored.

When TYPE=SINGLE and the value of LISTTYPE is not COMBO, ENTWIDTH specifies the number of bytes used for both the entry field and the space between the selection identifier and the selection text. The default width value is 2. The minimum width value is 1, which can be specified for any single-choice selection list. The maximum width value (when LISTTYPE is not COMBO) is 3, which can be specified for selection lists within a scrollable panel area. The width of 3 is provided for use when the number of CHOICE tags will exceed 99.

When LISTTYPE=COMBO, the maximum ENTWIDTH value is 2 bytes less than the SELWIDTH value.

Note: A width of 1 should only be used when the total number of CHOICE tags is less than 10. The conversion utility discards choices which cannot be selected with the specified entry width.

When TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR, the command area is used as the input field and the ENTWIDTH value is used only to determine the spacing between the selection identifier and the selection text. The maximum ENTWIDTH value for these types is 16.

REQUIRED=NO | YES
This attribute indicates if the field requires input.

If REQUIRED=YES is coded, a VER(variable,NONBLANK) statement will be built by ISPDTLC and placed in the)PROC section of the generated ISPF panel.

Note: This attribute is valid only when TYPE=SINGLE.

MSG=message-identifier
This attribute specifies the message that is displayed when the user does not choose a selection (defined with the REQUIRED attribute). If you do not specify a *message-identifier*, ISPF displays a default message.

If you specify the MSG attribute and REQUIRED=YES, a VER(variable,NONBLANK,MSG=message-identifier) statement is built by ISPDTLC and placed in the)PROC section of the generated ISPF panel. If you specify the MSG attribute and REQUIRED=NO (the default), the conversion utility issues a warning message.

FCHOICE=1 | 0
The FCHOICE attribute controls the starting choice number for TYPE=SINGLE, TYPE=MENU, TYPE=MODEL or TYPE=TUTOR. When FCHOICE=0, the first choice is the number 0.

Note: This attribute is valid only when TYPE=SINGLE, TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR.

AUTOTAB=YES | NO
When AUTOTAB=YES, the cursor moves to the next field capable of input when the user enters the last character in this field. If no other field capable of user input exists on the panel, the cursor returns to the beginning of this field.

The ISPF SKIP keyword is not supported when running in GUI mode.

Note: This attribute is valid only when TYPE=SINGLE.

DEPTH=n | *
This attribute defines the minimum size of a scrollable selection list. If DEPTH is not specified, the selection list will not be scrollable. If the DEPTH value is specified as "*", the conversion utility will reserve the remaining available panel depth. When EXTEND=OFF, the minimum depth is 2. When EXTEND=ON, the minimum depth is 1. The DEPTH attribute is ignored when LISTTYPE=COMBO.

EXTEND=OFF | ON | FORCE
This attribute defines the run-time display size for the scrollable list area. If EXTEND=ON is specified, the panel definition is expanded from the minimum DEPTH to the size of the logical screen. Only one EXTEND=ON

attribute value is allowed on a panel. The first tag (AREA, DA, GA, REGION, SELFLD) with EXTEND=ON is accepted; the EXTEND attribute on any subsequent tag is ignored.

If the EXTEND attribute is specified without the DEPTH attribute, a warning message is issued and the EXTEND attribute is ignored. The EXTEND attribute is ignored when LISTTYPE=COMBO.

If you intend to display the panels in a pop-up window, it is recommended that you code EXTEND=OFF.

If EXTEND=FORCE is specified within a horizontal area or region, the EXTEND(ON) keyword is added to the scrollable area attribute statement in the)ATTR panel section. The conversion utility issues a message to advise of a potential display error if other panel fields are formatted on or after the last defined line of the scrollable area.

TRAIL='trail-var-1 trail-var-2 ... trail-var-n'
This attribute specifies variable name(s) that the application uses to obtain the TRAIL information created by menu or model selection processing.

Each trail variable specified must follow the standard naming convention described in "Rules for Variable Names" on page 211.

Note: This attribute is valid only when TYPE=MENU or TYPE=MODEL.

CHOICECOLS=1 | n
This attribute specifies the number of columns to format with the CHOICE items. The default is 1. The CHOICECOLS attribute is ignored when LISTTYPE=COMBO.

CHOICEDEPTH=n | *
This attribute specifies the number of CHOICE entries to be placed in each column. The minimum CHOICEDEPTH value is 1. The normal maximum and default is the remaining panel depth. If the DEPTH attribute has been specified on the SELFLD tag, or an enclosing REGION or AREA tag, (and the corresponding tag attribute value for EXTEND is OFF) the most recently specified depth value is used as the maximum and default value. You may specify CHOICEDEPTH="*" which tells the conversion utility to calculate the column depth based on the total number of CHOICE tags and the number of columns specified by the CHOICECOLS attribute.

If more CHOICE entries are specified than can be formatted in the available number of columns specified by the CHOICECOLS attribute, the remaining CHOICE entries are placed in the rightmost (or only) available column for the current SELFLD tag. The CHOICEDEPTH attribute is ignored when LISTTYPE=COMBO.

CWIDTHS='w1 w2...wn'
This attribute specifies the number of bytes to be allocated for each column of CHOICE entries. The 'w1 w2...wn' notation provides the number of bytes for each column. You may use an asterisk or a number combined with an asterisk to specify a proportional allocation of column space. For example, the specification of '2* * 3*' for 3 columns would result in a space calculation based on 6 units, with 2 units allocated to column 1, 1 unit allocated to column 2, and 3 units allocated to column 3. If more columns have been specified by CHOICECOLS than are accounted for by CWIDTHS, the remaining space is divided evenly between the remaining columns. If CWIDTHS is not specified, the available formatting space is divided evenly based on the CHOICECOLS value. The CWIDTHS attribute is ignored when LISTTYPE=COMBO.

PAD=NULLS | USER | char | %varname
This attribute specifies the pad character for initializing the field. You can define this attribute as a variable name preceded by a "%".

Note: This attribute is valid only when TYPE=SINGLE.

PADC=NULLS | USER | char | %varname
This attribute specifies the conditional padding character to be used for initializing the field. You can define this attribute as a variable name preceded by a "%".

Note: This attribute is valid only when TYPE=SINGLE.

OUTLINE=NONE | L | R | O | U | BOX | %varname
This attribute provides for displaying lines around the field on a DBCS terminal. You can define this attribute as a variable name preceded by a "%".

Note: This attribute is valid only when TYPE=SINGLE.

SELMSG=selfld-msg-identifier
This attribute specifies the message that is displayed when an invalid single-choice entry is selected.

SELMSGU=selfld-msg-unavailable
This attribute specifies the message that is displayed when an unavailable single-choice entry is selected.

INIT=YES | NO | init-value
This attribute controls the single-choice and multi-choice selection field variables initialization in the panel)INIT section. When INIT = NO, the variables are not initialized to blank. When TYPE = SINGLE, you can alternatively provide a valid choice selection by specifying INIT = init-value.

CHOICE tag CHECKVAR processing can override the INIT value.

VERIFY=YES | NO
This attribute controls the single-choice verification and menu-choice, model-choice, or tutor-choice selection logic generation in the panel)PROC section. When TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR, VERIFY=NO bypasses the creation of the ZSEL statement. You can provide a replacement ZSEL statement with the <SOURCE> tag.

REFRESH=YES | NO
This attribute controls the creation of the REFRESH statement in the panel)REINIT section for multi-choice selection lists.

SELFMT=START | END
This attribute controls the placement of the choice selection character(s) within the width specified by ENTWIDTH. The default is to left justify the choice selection character(s).

Note: This attribute is valid only when TYPE=SINGLE, TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR.

CHKBOX=YES | NO
This attribute controls the creation of panel keywords that enable check boxes when running ISPF in GUI mode. The default value is YES.

The CHKBOX attribute is not valid and is ignored for single-choice, menu-choice, and model-choice selection lists.

If the conversion utility has been invoked with the NOGUI option, specifying CHKBOX=YES on the SELFLD tag will override the invocation option so that check-box controls will be generated.

ZGUI=YES | NO
This attribute controls the creation of the ″VGET (ZGUI)″ statement in the panel)INIT section for multi-choice selection lists that specify the ″&multipmt″ ENTITY as *field-prompt-text*.

CSRGRP=NO | YES | n
The CSRGRP attribute is valid only when TYPE=MULTI and CHKBOX=YES (either specified or defaulted). When CSRGRP=YES, the conversion utility generates a cursor group number to be used for this selection list. When CSRGRP=n, the number provided is used for the CHOICE fields within this SELFLD tag.

TSIZE='s1 s2...sn'
The TSIZE attribute provides the number of bytes to indent multiple lines of CHOICE text. Multiple TSIZE values can be used to provide unique indentation amounts for multiple column lists (when CHOICECOLS is greater than 1). If the number of TSIZE values is less than the number of columns, the last (or only) TSIZE value is used for the remaining columns. If more TSIZE values are supplied than there are columns of choices, the extra TSIZE values are ignored.

LISTTYPE=RADIO | LISTBOX | DDLIST | COMBO
This attribute controls the creation of panel keywords that cause single-choice selection lists to be displayed with radio buttons, or as a list box, drop-down list, or combination box when running ISPF in GUI mode.

When LISTTYPE=COMBO or LISTTYPE=DDLIST and the PANEL tag has specified TYPE=GUI, a single input field is placed in the panel)BODY section, and the DEPTH, EXTEND, CHOICECOLS, CHOICEDEPTH, and CWIDTHS attributes are ignored. The length of the input field is determined by the ENTWIDTH attribute. For combination boxes, you should consider placing a list of the valid possible choices in a help panel accessable through field-level help.

| **Note:** This attribute is valid only when TYPE=SINGLE.

LISTREF=list-name
This attribute is not used for LISTTYPE=RADIO. The *list-name* specifies the name for the)LIST section in the generated panel. The *list-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211. If you don't specify *list-name*, the default *list-name* is the *field-name* provided by the NAME attribute.

If a panel contains more than one SELFLD tag that has the same set of CHOICEs, the CHOICE tags can be provided within the first SELFLD tag definition, and then referenced in subsequent SELFLD tags by specifying the first SELFLD tag *list-name* as the LISTREF value of the subsequent SELFLD tag(s).

LISTDEPTH=n
The LISTDEPTH attribute is not used for LISTTYPE=RADIO. LISTDEPTH provides the number of panel lines to be used for the list box, drop-down list, or combination box.

When LISTTYPE=LISTBOX and you don't specify LISTDEPTH, the list box depth will be defaulted to use the number of panel lines formatted for a

numbered selection list, allowing for the horizontal scroll bar. If LISTDEPTH is specified, the value should be less than the number of lines formatted for a numbered selection list to allow for the horizontal scroll bar.

When LISTTYPE=DDLIST and you don't specify LISTDEPTH, the drop-down list depth is determined by ISPF when the panel is displayed. If LISTDEPTH is specified, the minimum LISTDEPTH value is 1. The normal maximum value is the remaining panel depth. If the DEPTH attribute has been specified on the SELFLD tag or on an enclosing REGION or AREA tag (and the corresponding tag attribute value for EXTEND is OFF), the most recently specified depth value is the maximum value.

DBALIGN=YES | NO | FIELD | FORCE
This attribute defines the DBALIGN value. DBALIGN is used only for DBCS language conversions when PMTLOC=ABOVE and the DBALIGN invocation option is specified.

When DBALIGN=YES, and the field-prompt-text starts with a DBCS character or a single-choice or multi-choice selection list definition does not include field-prompt-text, the entry field for the choice is shifted 1 position to the right.

When DBALIGN=NO, no alignment adjustment is made.

When DBALIGN=FIELD, the entry field is shifted but no adjustment is done for the prompt. The FORCE and FIELD values are useful when alignment is required with other SELFLD or DTAFLD tags.

When DBALIGN=FORCE, the entry field is shifted and the field-prompt-text is also adjusted to match even if the field-prompt-text starts with a single byte character.

NOSEL=no-selection-value
This attribute provides a value to be placed the CHECKVAR variable (specified by the CHOICE tag) when no selection is chosen from the available list.

If REQUIRED=YES is specified, a message is issued and NOSEL is ignored.

If no CHOICE tag specifies a CHECKVAR attribute, the NOSEL attribute is ignored.

Note: This attribute is valid only when TYPE=SINGLE.

SELDEFAULT=x
This attribute is used to provide a default choice selection when TYPE=SINGLE, MENU, MODEL, or TUTOR. The value x must be a valid choice selection. If no selection is made by the user, the default value is returned to the application.

PMTSKIP=NO | YES
This attribute is used for horizontal formatting of input fields. When PMTSKIP=YES, and the previous DTAFLD definition includes the NOENDATTR attribute, the cursor will move past the prompt text to the input field when the user enters the last character in the previous field. If there is no other input field on the panel, the cursor returns to the first input field on the panel. The ISPF SKIP keyword is not supported in GUI mode.

FLDTYPE=CUA | ISPF
This attribute defines the attribute type to be applied to the selection entry field when LISTTYPE is LISTBOX, DDLIST, or COMBO. TYPE=CUA, the default, causes the field to display using the standard CUA attribute. When

FLDTYPE=ISPF, a non-CUA attribute is generated and you may specify the color, intensity and highlighting with the COLOR, INTENS and HILITE attributes. These attributes are not valid when FLDTYPE=CUA.

COLOR=WHITE | RED | BLUE | GREEN | PINK | YELLOW | TURQ | %varname
This attribute specifies the color of the field. You can define this attribute as a variable name preceded by a "%".

INTENS=HIGH | LOW | NON | %varname
This attribute defines the intensity of a field. You can define this attribute as a variable name preceded by a "%".

HILITE=USCORE | BLINK | REVERSE | %varname
This attribute specifies the extended highlighting attribute of the field. You can define this attribute as a variable name preceded by "%".

SELCHECK = NO | YES
This attribute is used with menu-choice selection to specify that panel logic is to be included in selection processing to check for valid selection entries. For example, a message is issued if a period (.) or a period followed by data (.xxx) is entered as a selection choice.

Note: This attribute is valid only when TYPE=MENU.

VARDCL=YES | NO
When VARDCL=NO the field name is not checked to the declared variable information provided with the VARCLASS and VARDCL tags.

Note: This attribute is only valid when TYPE=SINGLE.

field-prompt-text
This is the prompt text for the selection field. The prompt text can appear in front of or above the field, based on the value assigned to the PMTLOC attribute.

Multi-choice selections are displayed as check boxes when running in GUI mode. To support both host and workstation forms of multi-choice prompt text, a special pre-defined ENTITY name of "&multipmt" may be specified as the *field-prompt-text*. When the panel is displayed, the *field-prompt-text* will be

```
Enter "/" to select option
```

(or its translated equivalent) for host display or

```
Check box to select option
```

(or its translated equivalent) for workstation display. The panel definition should specify a PMTWIDTH value large enough to format the prompt as a single line. If there is insufficient space to present the entire *field-prompt-text*, it will be truncated to fit the available space.

Description

The SELFLD tag defines a selection field that includes a list of choices. CHOICE tags coded within the SELFLD definition define the choices for the selection field.

The TYPE attribute of the SELFLD tag determines how the choices appear. If TYPE=SINGLE, the SELFLD NAME attribute is used as the selection input field. If TYPE=MULTI, the CHOICE NAME attribute is used as the selection input field for each choice. If TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR, the command line is used as the selection input field.

When a selection list is formatted as a scrollable list:

- The multi-choice list entry field scrolls with the choice descriptions.
- The single-choice entry field is formatted beside the choice list and remains visible when the choice descriptions scroll.
- Choice descriptions that are formatted in multiple columns (CHOICECOLS and CHOICEDEPTH attributes specified) result in a separate scrollable area for each column.

The)LIST section is added to the panel if you specify:

- The LISTREF attribute
- A scrollable selection list (DEPTH is provided)
- The SELFLD tag within a scrollable AREA or REGION
- A multiple column selection list (CHOICECOLS > 1)
- LISTTYPE=COMBO
- TYPE=GUI on the PANEL tag.

Note: If you specify the CMDAREA tag within your DTL source file, it must appear before the SELFLD tag when TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR and CHECKVAR or UNAVAIL attributes are specified on nested CHOICE tags.

If you specify the CMDAREA tag within your DTL source file, it must appear before the SELFLD tag when DEPTH=* is specified. The SELFLD tag DEPTH may have to be adjusted to allow for additional lines which result from tags present within the panel definition following the end SELFLD tag.

Conditions

- The SELFLD tag requires an end tag.
- You must code the SELFLD tag within an AREA, DTACOL, REGION, or PANEL definition. See “AREA (Area)” on page 223, “DTACOL (Data Column)” on page 306, “REGION (Region)” on page 445, and “PANEL (Panel)” on page 415 for descriptions of these tags.
- Single-choice selection fields (the default TYPE value) should have an associated VARDCL definition for the *field-name* specified with the NAME attribute. See “VARDCL (Variable Declaration)” on page 486 for a complete description of this tag.
- If both PAD and PADC have been specified, PAD is ignored and PADC is used.
- When a “%varname” notation is found on any of the attributes that allow a variable name, the “%varname” entry must follow the standard naming convention described in “Rules for “%variable” Names” on page 211.
- You should code a CMDAREA on any panel that contains a SELFLD definition that specifies TYPE=MENU, TYPE=MODEL, or TYPE=TUTOR. If you do not include the CMDAREA tag, the conversion utility inserts one and issues a message, unless the PANEL tag specifies CMDLINE=NO.
- Only one menu-choice or model-choice list is formatted for any panel. If multiple menu-choice or model-choice lists are specified, the first one will be formatted as a menu; subsequent menu-choice or model-choice lists will be formatted as single-choice lists.

Nested Tags

You can code the following tags within a SELFLD definition:

Tag	Name	Usage	Page	Required
CHDIV	Choice Divider	Multiple	242	No
CHOICE	Selection choice	Multiple	261	No
COMMENT	Comment	Multiple	282	No
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SOURCE	Source	Multiple	472	No

Example

The following application panel markup contains two selection fields. The first selection field is a single-choice selection field with the prompt text located before the selection field. The single-choice selection field can be preselected depending on the value assigned to the variable *card*.

The second selection field is a multiple-choice selection field with the prompt text located above the selection field. Choices within this field may be preselected depending on the value assigned to the CHECKVAR attribute variable specified on the respective CHOICE tags.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 sysem>
  <!entity sampabc system>)>
&sampvar1;

<PANEL NAME=selfld3 KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
<TOPINST>Type in patron's name and card number (if applicable).
<TOPINST>Then select an action bar choice.
<AREA>
  <DTAFLD DATAVAR=curdate PMTWIDTH=12 ENTWIDTH=8 USAGE=out>Date
  <DTAFLD DATAVAR=cardno PMTWIDTH=12 ENTWIDTH=7 DESWIDTH=25>Card No
    <DTAFLDD>(A 7-digit number)
  <DTAFLD DATAVAR=name PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25>Name
    <DTAFLDD>(Last, First, M.I.)
  <DTAFLD DATAVAR=address PMTWIDTH=12 ENTWIDTH=25>Address
  <DIVIDER>
  <REGION DIR=horiz>
  <SELFLD NAME=cardsel PMTWIDTH=30 SELWIDTH=40
          entwidth=1 required=yes autotab=yes>
          Choose one of the following
    <CHOICE CHECKVAR=card MATCH=new>New
    <CHOICE CHECKVAR=card MATCH=renew>Renewal
    <CHOICE CHECKVAR=card MATCH=replace>Replacement
  </SELFLD>
  <SELFLD TYPE=multi PMTWIDTH=30 SELWIDTH=36
          depth=5 init=no>
          Check valid branches
    <CHOICE NAME=north HELP=nthhlp CHECKVAR=nth>North Branch
    <CHOICE NAME=south HELP=sthhlp CHECKVAR=sth>South Branch
    <CHOICE NAME=east HELP=esthlp CHECKVAR=est>East Branch
    <CHOICE NAME=west HELP=wsthlp CHECKVAR=wst>West Branch
    <CHOICE NAME=city HELP=ctyhlp CHECKVAR=cty>City Branch
    <CHOICE NAME=cnty HELP=cnthlp CHECKVAR=cnt>County Branch
```

```
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

Figure 163 shows the formatted result.

```
  File  Search  Help
------------------------------------------------------------------------
                        Library Card Registration

Type in patron's name and card number (if applicable).

Then select an action bar choice.

Date . . . : ________
Card No  . . _______  (A 7-digit number)
Name . . . . _____________________  (Last, First, M.I.)
Address  . . _____________________

                                      Check valid branches
Choose one of the following           #SAREA37                          #
_  1. New                             #                                 #
   2. Renewal                         #                                 #
   3. Replacement                     #                                 #
                                      #                                 #
Enter a command ===> ___________________________________________________
 F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

The contents of the scrollable area are as follows:

```
         )AREA SAREA37
```

```
_  North Branch
_  South Branch
_  East Branch
_  West Branch
_  City Branch
_  County Branch
```

```
         )AREA SAREA37
```

Figure 163. Selection Fields

The following example illustrates the creation of an ISPF selection menu. The FCHOICE attribute specifies that the first selection number is 0. The choice selection for Exit is specified on the CHOICE tag. The ACTION tag for the Exit choice selection specifies both the RUN and TYPE attributes because RUN is required on the ACTION tag and TYPE is necessary to specify the ISPF selection for the generated ZSEL panel statement.

```
<!doctype dm system ()>
<!--  Sample selection menu -->
<varclass name=vc1 type='char 80'>
  <xlatl format=upper>
  </xlatl>
```

```
<varlist>
  <vardcl name=zcmd varclass=vc1>
</varlist>

<panel name=selfld2 menu keylist=keylxmp>Sample Selection Panel
  <topinst>This is a selection panel.
  <selfld type=menu   pmtloc=before fchoice=0 trail=nextsel
          selwidth=40 pmtwidth=10>Select an option
    <choice checkvar=xtest1 match=a>
            Selection #0 (Command Selch0)
      <action run=Selch0>
    <choice checkvar=xtest1 match=b>
            Selection #1 (Command Selch1)
      <action run=Selch1 parm='1 2 3 4'
       passlib newpool suspend>
    <choice checkvar=xtest1 match=c>
            Selection #2 (Command Selch2)
      <action run=Selch2 parm=1234>
    <choice checkvar=xtest1 match=d>
            Selection #3 (Command Selch3)
      <action run=Selch3 parm=abcd>
    <choice checkvar=xtest1 match=e>
            Selection #4 (Command Selch4)
      <action run=Selch4 parm='a b c d'>
    <choice selchar=X>
            Exit
      <action run=exit type=exit>
  </selfld>
  <cmdarea>
</panel>
```

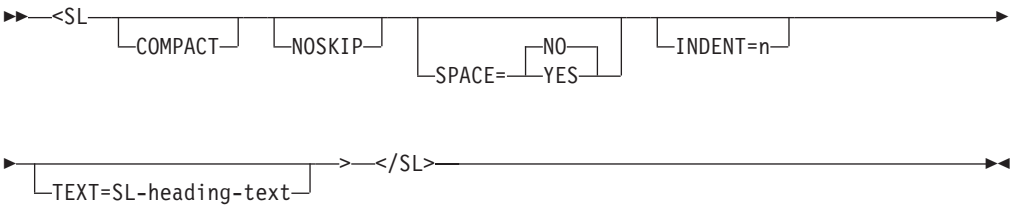
Figure 164 shows the formatted result.

```
                          Sample Selection Panel

This is a selection panel.

Select an
option . . 0  Selection #0 (Command Selch0)
           1  Selection #1 (Command Selch1)
           2  Selection #2 (Command Selch2)
           3  Selection #3 (Command Selch3)
           4  Selection #4 (Command Selch4)
           X  Exit









Option ===> ____________________________________________________________
 F1=Help        F3=Exit        F5=Display     F6=Keyshelp   F10=Actions
F12=Cancel
```

Figure 164. Selection Menu

SL (Simple List)

The SL tag defines a simple list of items within an information region.



COMPACT
This attribute causes the list to be formatted without a blank line between the list items.

NOSKIP
This attribute causes the list to format without creating a blank line before the first line of the list.

SPACE=NO | YES
The SPACE attribute controls the indentation space for the list item. When the SPACE attribute is not specified on the LI tag, the SPACE attribute from the SL tag is used to set the indentation space for the nested LI tag *item-text*.

When SPACE=YES, the indentation is set to 3 spaces. When SPACE=NO (or SPACE is not specified), the indentation is set to 4 spaces.

The SPACE attribute can be used to control the alignment of list items when the first word of some list items is a DBCS word preceded by a shift-out character and the first word of other list items is a SBCS word.

INDENT=n
This attribute specifies that the list be indented from the current left margin.

TEXT=SL-heading-text
This attribute causes the list to format with a heading line containing the *SL-heading-text*.

Description

The SL tag defines a simple list of items within an information region.

Simple lists are indented lists, with no bullets, dashes, or hyphens preceding the list items. Nested lists indent four spaces to the right of the left margin of the list that contains them.

Note: The SPACE attribute does not affect the indentation of nested lists.

The conversion utility adds a blank line before the first item in the list.

Use the LI tag to denote each list item. See "LI (List Item)" on page 364 for more information on the LI tag.

Conditions

- The SL tag requires an end tag.
- You must code the SL tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.

Nested Tags

You can code the following tags within the SL tag:

Tag	Name	Usage	Page	Required
LI	List item	Multiple	364	No
LP	List part	Multiple	370	No

Example

The following help panel markup contains two simple lists. The second simple list is compact, and is nested within the first list. Figure 165 shows the formatted result.

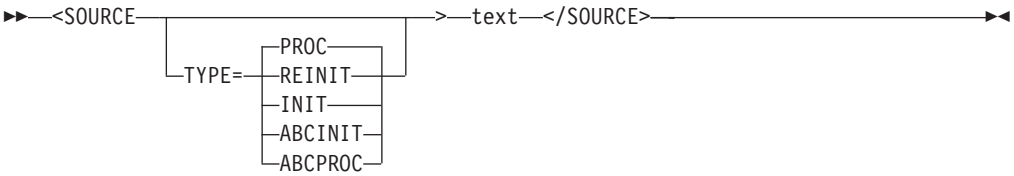
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=sl WIDTH=40 DEPTH=22>Help for ShelfBrowse
<AREA>
<INFO>
  <P>Using ShelfBrowse, you can locate the following items:
  <SL>
    <LI>Audiotapes
    <LI>Books
    <LI>Periodicals
      <SL COMPACT>
        <LI>Newspapers
        <LI>Magazines
      </SL>
    <LI>Reference material
    <LI>Videotapes
</SL>
</INFO>
</AREA>
</HELP>
```

```
          Help for ShelfBrowse

 Using ShelfBrowse, you can locate the
 following items:

     Audiotapes

     Books

     Periodicals

         Newspapers
         Magazines

     Reference material

     Videotapes


  F1=Help      F3=Exit      F5=Exhelp
  F6=Keyshelp  F7=PrvTopic  F8=NxtTopic
 F10=PrvPage  F11=NxtPage  F12=Cancel
```

Figure 165. Simple List

SOURCE (Source)

The SOURCE tag defines ISPF panel logic statements within an application panel.



TYPE=PROC | REINIT | INIT | ABCINIT | ABCPROC
This attribute specifies the panel section that will be updated with the SOURCE tag *text*.

text
This is the unformatted ISPF panel statement.

Description

The SOURCE tag defines ISPF panel statements within an application panel.

Lines of text from a SOURCE tag that follows an AREA, CHOICE, DA, DTACOL, DTAFLD, HELP, LSTCOL, LSTFLD, LSTGRP, PANEL, REGION, or SELFLD tag are added to the)INIT,)REINIT, or)PROC panel section when encountered in the DTL source file.

For example, if a SOURCE tag follows the DTAFLD tag, any logic or other entries normally generated by DTAFLD would be completed before the lines within SOURCE are added.

The use of a SOURCE tag within a SELFLD tag results in the placement of the SOURCE tag lines after any logic created by the previous CHOICE tag. Additional)INIT,)REINIT, or)PROC section entries may be added when the end SELFLD tag is processed. You can control the placement of the SOURCE tag entries by nesting the SELFLD tag definition within a DTACOL tag, and placing the SOURCE tag definition either before or after the SELFLD tag definition.

Lines of text from a SOURCE tag within an action bar definition are added to:
-)ABCINIT *following* all other generated statements for that PDC tag.
-)ABCPROC *before* any other generated statements for that PDC tag.

SOURCE tags within an action bar definition must specify the TYPE as ABCINIT or ABCPROC. SOURCE tags that follow the other listed tags cannot specify TYPE as ABCINIT or ABCPROC.

When the SOURCE tag is coded within a GENERATE tag, the TYPE attribute is ignored. TYPE is automatically determined from the placement of the GENERATE tag within the DTL source file.

If the length of any line exceeds the record length of the output panel file, the conversion utility truncates the line and issues a warning message.

Text found between the SOURCE and SOURCE end tags is placed in the specified panel section as coded; that is, no formatting except entity substitution is performed. To refer to an entity within <SOURCE> tag text, the entity name is

preceded by a percent (%) instead of an ampersand (&). Using the percent (%) sign avoids conflict with variable names. A valid percent sign can be specified as "%amp;" to avoid an "entity not found" message. For example, you would refer to the TSO command "%xyz" as "%amp;xyz".

Conditions

- The SOURCE tag requires an end tag.
- You must code the SOURCE tag within an ABC, AREA, CHOICE, DA, DTACOL, DTAFLD, GENERATE, HELP, INFO, LSTCOL, LSTFLD, LSTGRP, PANEL, PDC, REGION, or SELFLD tag definition.

Nested Tags

None.

Example

```
<!DOCTYPE DM SYSTEM(
  <!entity sampvar1 sysem>
  <!entity sampabc system>)>
&sampvar1;

<PANEL NAME=source1 KEYLIST=keylxmp>Library Card Registration
<AB>
&sampabc;
</AB>
<TOPINST>Type in patron's name and card number (if applicable)
<TOPINST>Then select an action bar choice.
<AREA>
    <DTACOL PMTWIDTH=12 ENTWIDTH=25 DESWIDTH=25 SELWIDTH=25>
    <DTAFLD DATAVAR=curdate USAGE=out ENTWIDTH=8>Date
    <DTAFLD DATAVAR=cardno ENTWIDTH=7>Card No.
      <DTAFLDD>(A 7-digit number)
    <DTAFLD DATAVAR=name>Name
      <DTAFLDD>(Last, First, M.I.)
    <DTAFLD DATAVAR=address>Address
   </DTACOL>
  <DIVIDER>
  <REGION DIR=horiz>
  <SELFLD NAME=cardsel PMTWIDTH=30 SELWIDTH=38>Choose
  one of the following
    <CHOICE CHECKVAR=card MATCH=new>New
    <CHOICE CHECKVAR=card MATCH=renew>Renewal
    <CHOICE CHECKVAR=card MATCH=replace>Replacement
    <SOURCE TYPE=proc>
    if (&cardsel = 1)
      VER (&name,nb)
      VER (&address,nb)
    </SOURCE>
  </SELFLD>
  <SELFLD TYPE=multi PMTWIDTH=30 SELWIDTH=25>Check valid branches
    <CHOICE NAME=north HELP=nthhlp CHECKVAR=nth>North Branch
    <CHOICE NAME=south HELP=sthhlp CHECKVAR=sth>South Branch
    <CHOICE NAME=east HELP=esthlp CHECKVAR=est>East Branch
    <CHOICE NAME=west HELP=wsthlp CHECKVAR=wst>West Branch
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

T (Truncation)

The T tag designates the minimum command name that the user must enter to issue a command.

Description

You must code the T tag within the *external-command-name* of the CMD tag. For example, if the following command is coded in an application command table, the user could enter `com`, `comp`, `compa`, `compar`, or `compare` to run the command.

```
<cmd name=compare>com<t>pare
```

The command name must be at least 2 bytes.

At run time, ISPF runs the first valid command in the command table that matches the character string entered in the command area.

You should be careful to avoid specifying values that conflict with other commands. For example:

```
<cmd name=compare>co<t>mpare
<cmd name=copy>co<t>py
```

In this situation, if the user enters `co` as a command, ISPF will run the COMPARE command.

Conditions

- You must code the T tag within the *external-command-name* of a CMD definition. See "CMD (Command Definition)" on page 268 for a complete description of this tag.

Nested Tags

None.

Example

The following source file markup contains a command table. The commands DELETE and UPDATE have truncation definitions that allow the user to enter "del" and "upd", respectively, as the minimum command name.

```
<!DOCTYPE DM SYSTEM>

<CMDTBL APPLID=conv>
  <CMD NAME=update>Upd<T>ate
    <CMDACT ACTION='alias add'>
  <CMD NAME=add>Add
    <CMDACT ACTION=setverb>
  <CMD NAME=delete>Del<T>ete
    <CMDACT ACTION=passthru>
  <CMD NAME=search>Search
    <CMDACT ACTION=passthru>
</CMDTBL>
```

The following table shows the resultant ISPF application command table.

Table 6. ISPF Application Command Table

ZCTVERB	ZCTTRUNC	ZCTACT
UPDATE	3	ALIAS ADD
ADD	0	SETVERB
DELETE	3	PASSTHRU
SEARCH	0	PASSTHRU

TEXTLINE (Text Line)

The TEXTLINE tag generates a single line of text to replace the regular tag text for the HELP and PANEL tags.

Description

The TEXTLINE tag encloses one or more TEXTSEG tags, used to define the parts or segments of the replacement text. Text defined by the TEXTSEG tag(s) is accumulated in a left to right order. The resulting text is used to to create or replace the text portion of the HELP or PANEL tag definition.

Conditions

- The TEXTLINE tag requires an end tag.
- You must code the TEXTLINE tag within a HELP or PANEL tag definition.

Nested Tags

You can code the following tags within a FIG definition:

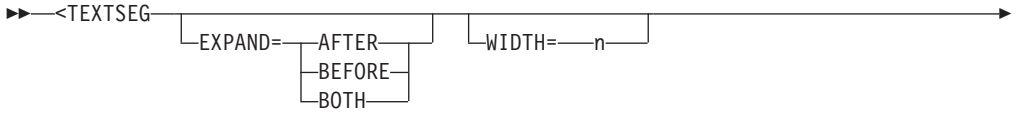
Tag	Name	Usage	Page	Required
DTAFLD	Data Field	Multiple	312	No
TEXTSEG	Text Segment	Multiple	475	Yes

Example

See the example for “TEXTSEG (Text Segment)”.

TEXTSEG (Text Segment)

The TEXTSEG tag creates a text segment to be accumulated for the replacement text line created by the TEXTLINE tag.



EXPAND=ABOVE | BEFORE | BOTH
This attribute specifies whether expand control is added to the provided text. Expand characters are obtained from the HELP or PANEL tag definition, if available. If no expand character(s) have been specified on those tags, the conversion utility generates the necessary character. You may place the expand control before, after, or both before and after the text.

WIDTH=n
This attribute specifies the number of bytes to reserve for the text. The default is to not allow space beyond the actual text length.

Text
This is the text of the segment.

Description

The TEXTSEG tag defines a part or segment of a replacement text line. When multiple TEXTSEG tags are present within the TEXTLINE definition, the replacement text line is created from left to right in the order the TEXTSEG tags are coded.

Conditions

- You must code the TEXTSEG tag within a TEXTLINE tag definition.
- If the EXPAND attribute is not specified and the resulting replacment text is less than the panel width, the text will be centered as the panel title.

Nested Tags

You can code the following tags within a FIG definition:

Tag	Name	Usage	Page	Required
HP	Highlighted Phrase	Multiple	354	No

Example

The follwing example uses the TEXTLINE and TEXTSEG tags to create a special panel title that includes the system time and date. Because the EXPAND attribute is specified in the second TEXTSEG tag, the resulting title replacement text has the time and date fields placed at the left and right panel border.

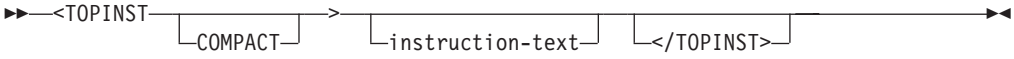
```
<!doctype dm system ()>
<!--   Sample selection menu -->

<varclass name=vc1 type='char 80'>
   <xlatl format=upper>
   </xlatl>

<varlist>
  <vardcl name=zcmd varclass=vc1>
</varlist>

<panel name=textseg1 menu keylist=keylxmp>

   <textline>
     <textseg>&ztime
     <textseg expand=both>
```

```
          Sample Selection Panel with TEXTLINE tag
     <textseg>&zdate(8)
   </textline>

  <topinst>This is a selection panel.
  <selfld type=menu   pmtloc=before fchoice=0 trail=nextsel
          selwidth=40 pmtwidth=10>Select an option
    <choice checkvar=xtest1 match=a>
                Selection #0 (Command Selch0)
      <action run=Selch0>
    <choice checkvar=xtext1 match=b>
                Selection #1 (Command Selch1)
      <action run=Selch1 parm='1 2 3 4'>
       passlib newpool suspend
    <choice checkvar=xtest1 match=c>
                Selection #2 (Command Selch2)
      <action run=Selch2 parm=1234>
    <choice checkvar=xtest1 match=d>
                Selection #3 (Command Selch3)
      <action run=Selch3 parm=abcd>
    <choice checkvar=xtest1 match=e>
                Selection #4 (Command Selch4)
      <action run=Selch4 parm='a b c d'>
    <chdiv>
    <choice selchar=x>
            Exit
      <action run=exit type=exit>
  </selfld>
  <cmdarea>
</panel>
```

```
07:30           Sample Selection Panel with TEXTLINE tag          99/12/15
Option ===> _

This is a selection panel.

Select an
option . . 0  Selection #0 (Command Selch0)
           1  Selection #1 (Command Selch1)
           2  Selection #2 (Command Selch2)
           3  Selection #3 (Command Selch3)
           4  Selection #4 (Command Selch4)

           X  Exit
```

TOPINST (Top Instruction)

The TOPINST tag defines top instructions for an application panel.



COMPACT
This attribute causes the top instruction to format without a blank line after the text.

instruction-text
This is the text of the top instruction. The *instruction-text* must fit in the remaining panel depth.

Description

The TOPINST tag defines top instructions for an application panel. The *instruction-text* formats as a paragraph based on the width of the application panel. You can code multiple paragraphs of instruction text by using a new top instruction tag for each new paragraph.

If the COMPACT attribute is not specified, the conversion utility inserts a blank line after the top instruction text.

Conditions

- You must code the TOPINST within a PANEL definition. See "PANEL (Panel)" on page 415 for a complete description of this tag.
- You cannot code a TOPINST tag within an AREA definition. If you define an area for the panel, code the TOPINST tag before the AREA start tag.

Nested Tags

You can code the following tag within a TOPINST definition:

Tag	Name	Usage	Page	Required
HP	Highlighted phrase	Multiple	354	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No

Example

The following application panel markup contains top instructions. Figure 166 on page 479 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=selcls TYPE='CHAR 2'>
<VARLIST>
  <VARDCL NAME=loc  VARCLASS=selcls>
  <VARDCL NAME=mode VARCLASS=selcls>
</VARLIST>

<PANEL NAME=topinst HELP=trvlhlp WIDTH=60 DEPTH=22 KEYLIST=keylxmp>
Dream Vacation Guide
<AB>
  <ABC>File
    <PDC>Add Entry
        <ACTION RUN=add>
    <PDC>Delete Entry
        <ACTION RUN=delete>
    <PDC>Update Entry
        <ACTION RUN=update>
    <PDC>Exit
        <ACTION RUN=exit>
  <ABC>Help
    <PDC>Extended Help...
```

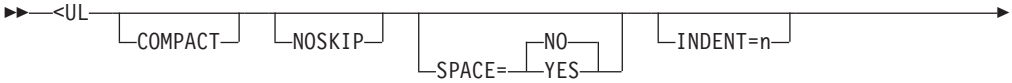
```
        <ACTION RUN=exhelp>
    <PDC>Keys Help...
        <ACTION RUN=keyshelp>
</AB>
<TOPINST>Choose one of the following exotic locations and
your preferred mode of travel, then press Enter.
<AREA>
  <REGION DIR=horiz>
  <SELFLD NAME=loc PMTWIDTH=23 SELWIDTH=25>Exotic Location:
    <CHOICE>Athens, GA
    <CHOICE>Berlin, CT
    <CHOICE>Cairo, IL
    <CHOICE>Lizard Lick, NC
    <CHOICE>Paris, TX
    <CHOICE>Rome, NY
    <CHOICE>Venice, FL
  </SELFLD>
  <DIVIDER>
  <SELFLD NAME=mode PMTWIDTH=25 SELWIDTH=25>Travel Mode:
    <CHOICE>Boxcar
    <CHOICE>Hitchhike
    <CHOICE>Mule
  </SELFLD>
  </REGION>
</AREA>
<CMDAREA>
</PANEL>
```

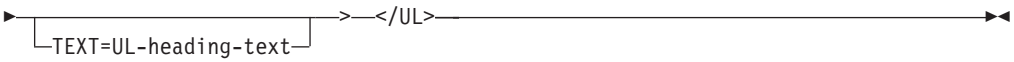
```
  File  Help
-------------------------------------------------------
                   Dream Vacation Guide

Choose one of the following exotic locations and your
preferred mode of travel, then press Enter.

Exotic Location:            Travel Mode:
__  1.  Athens, GA          __  1.  Boxcar
    2.  Berlin, CT              2.  Hitchhike
    3.  Cairo, IL               3.  Mule
    4.  Lizard Lick, NC
    5.  Paris, TX
    6.  Rome, NY
    7.  Venice, FL




Command ===>_____________________________________________
 F1=Help     F2=Split       F3=Exit     F6=Keyshelp
 F9=Swap    F12=Cancel
```

Figure 166. Top Instructions

UL (Unordered List)

The UL tag defines an unordered list of items within an information region.





COMPACT
This attribute causes the list to be formatted without a blank line between the list items.

NOSKIP
This attribute causes the list to format without creating a blank line before the first line of the list.

SPACE=NO | YES
The SPACE attribute controls the indentation space for the list item. When the SPACE attribute is not specified on the LI tag, the SPACE attribute from the UL tag is used to set the indentation space for the nested LI tag *item-text*.

When SPACE=YES, the indentation is set to 3 spaces. When SPACE=NO (or SPACE is not specified), the indentation is set to 4 spaces.

The SPACE attribute can be used to control the alignment of list items when the first word of some list items is a DBCS word preceded by a shift-out character and the first word of other list items is a SBCS word.

INDENT=n
This attribute specifies that the list be indented from the current left margin.

TEXT=UL-heading-text
This attribute causes the list to format with a heading line containing the *UL-heading-text*.

Description

The UL tag defines an unordered list of items within an information region. Unordered lists format as indented lists, with the list item identifier at the left margin. Nested lists indent four spaces to the right of the left margin of the list that contains them.

Note: The SPACE attribute does not affect the indentation of nested lists.

The conversion utility adds a blank line before the first item in the list. There are three levels of item identifiers: bullets (o), hyphens (-), and dashes (--). Each level is used successively when you nest unordered lists.

Panels formatted with the DBCS option use an uppercase 'O' as the bullet character.

Use the LI tag to denote each list item. See "LI (List Item)" on page 364 for more information on the LI tag.

Conditions

- The UL tag requires an end tag.
- You must code the UL tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.

Nested Tags

You can code the following tags within a UL definition:

Tag	Name	Usage	Page	Required
LI	List item	Multiple	364	No
LP	List part	Multiple	370	No

Example

The following help panel markup contains two unordered lists. The second unordered list is nested within the second list item of the first unordered list. Figure 167 on page 482 shows the formatted result.

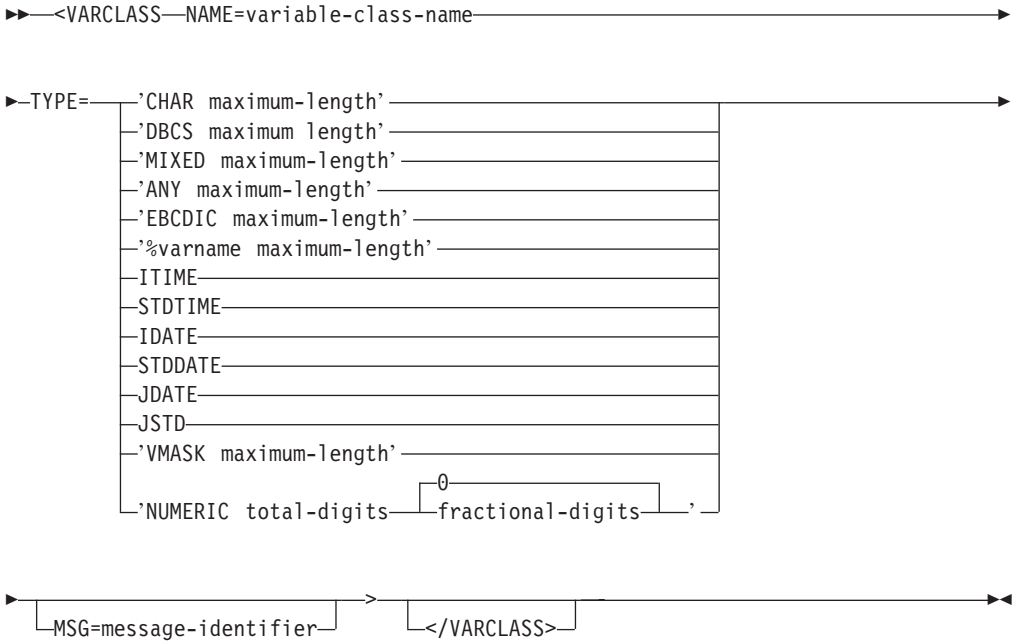
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=ul DEPTH=22>Help for Reference Section
<AREA>
<INFO>
  <P>Learn everything about anything,
  and more, in our Reference section.
  Our Reference section includes:
  <UL>
    <LI>Atlases
    <LI>Dictionaries
      <UL COMPACT>
        <LI>English
        <LI>Other languages
      </UL>
    <LI>Encyclopedias
    <LI>How-to books
    <LI>Magazines and periodicals
  </UL>
</INFO>
</AREA>
</HELP>
```

```
           Help for Reference Section

Learn everything about anything, and more, in
our Reference section. Our Reference section
includes:

o   Atlases

o   Dictionaries

    -   English
    -   Other languages

o   Encyclopedias

o   How-to books

o   Magazines and periodicals

 F1=Help         F3=Exit         F5=Exhelp
 F6=Keyshelp     F7=PrvTopic     F8=NxtTopic
F10=PrvPage     F11=NxtPage     F12=Cancel
```

Figure 167. Unordered List

VARCLASS (Variable Class)

The VARCLASS tag defines information related to a class of variables.



NAME=variable-class-name
This attribute specifies the name of this variable class.

The *variable-class-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

TYPE=type of data

This attribute specifies the data type and display length characteristics for variables that refer to the variable class.

For data fields and list columns, the conversion utility uses the lengths specified in this attribute when CHOFLD or DTAFLD ENTWIDTH or LSTCOL COLWIDTH attributes cannot otherwise be determined. The lengths specified control the width of the data field in the panel.

The allowable TYPE values are:

'CHAR maximum-length'
This specifies a character string for which the maximum length, in bytes, is given by *maximum-length*.

'DBCS maximum-length'
This is a double-byte character string for which the maximum length, in bytes, is given by *maximum-length*. The maximum length must an even number.

'MIXED maximum-length'
This specifies a character string containing single-byte characters, double-byte characters, or both for which the maximum length, in bytes, is given by *maximum-length*. Strings of DBCS characters are delimited by shift-out (SO) and shift-in (SI) codes.

'ANY maximum-length'
This attribute is processed by the conversion utility as TYPE=MIXED.

'EBCDIC maximum-length'
This specifies a character string containing only single-byte characters for which the maximum length, in bytes, is given by *maximum-length*.

'%varname maximum-length'
This specifies a variable name will be used to define the type of character string. The maximum length, in bytes, is given by *maximum-length*. It is the responsibility of the application developer to ensure that **%varname** contains a valid TYPE value prior to attempting to display the panel.

ITIME
The conversion utility will add a ″VEDIT (*variable*)″ statement to the)PROC section of the panel for variables which are related to this VARCLASS. The default length value of ITIME is set by the conversion utility to 5.

STDTIME
The conversion utility will add a ″VEDIT (*variable*)″ statement to the)PROC section of the panel for variables which are related to this VARCLASS. The default length value of STDTIME is set by the conversion utility to 8.

IDATE
The conversion utility will add a ″VEDIT (*variable*)″ statement to the)PROC section of the panel for variables which are related to this VARCLASS. The default length value of IDATE is set by the conversion utility to 8.

STDDATE
The conversion utility will add a ″VEDIT (*variable*)″ statement to the)PROC section of the panel for variables which are related to this VARCLASS. The default length value of STDDATE is set by the conversion utility to 10.

JDATE
This attribute is supported as an ISPF extension to the Dialog Tag Language. The conversion utility will add a ″VEDIT (*variable*)″ statement to the)PROC section of the panel for variables which are related to this VARCLASS. The default length value of JDATE is set by the conversion utility to 6.

JSTD
This attribute is supported as an ISPF extension to the Dialog Tag Language. The conversion utility will add a ″VEDIT (*variable*)″ statement to the)PROC section of the panel for variables which are related to this VARCLASS. The default length value of JSTD is set by the conversion utility to 8.

'VMASK maximum-length'
This attribute is supported as an ISPF extension to the Dialog Tag Language. the VMASK attribute is provided to support the user mask option the the ISPF VMASK service. The *maximum-length* value is limited to the ISPF maximum of 20. The conversion utility will add a ″VEDIT (*variable*)″ statement to the)PROC section of the panel for variables which are related to this VARCLASS.

'NUMERIC total-digits 0 | fractional-digits'
This attribute allows you to check to see if the user has entered a valid number. A valid number can include thousands separators, a decimal separator, and a sign. The conversion utility builds the VER(variable ENUM) statement to perform numeric validation. The value specified for *total-digits* must not be greater than 16.

The *total-digits* and *fractional-digits* are used to determine a *maximum-length* value which is used for field entry width, if necessary, in DTAFLD and LSTCOL processing. For example, 'NUMERIC 8 2' defines a width of 11, composed of 8 possible digits, a decimal point, a thousands separator, and a leading sign.

Note: ISPF does not check to verify proper positioning of the decimal point. Refer to the discussion on VER(variable ENUM) in the *ISPF User's Guide* for more information.

MSG=message-identifier
This attribute indicates the default message to be displayed if the variable fails any of the enclosed checks. See "MSG (Message)" on page 393 for information on creating messages.

Description

The VARCLASS tag defines information related to a class of variables. You can group validation and translation checks you want ISPF to perform within one VARCLASS definition. You point to the VARCLASS definition from one or more VARDCL tags you code within the VARLIST definition.

Note: The ISPF Dialog Tag Language conversion utility does not require that you code the VARCLASS, VARDCL, or VARLIST tags for a successful generation of a panel, command table, or message member that includes variables. If the conversion utility finds a variable that does not have an associated VARDCL definition, it issues a warning message.

The use of the VARCLASS, VARDCL, and VARLIST tags is required if you want to use the facilities provided by the CHECKL and XLATL tags.

Conditions

- You cannot code the VARCLASS tag within any other tag definition.
- You must code the VARCLASS tag before any other tag within the source file that refers to it.
- Within the variable class definition, you must code any and all XLATL tags before any CHECKL tags.

Nested Tags

You can code the following tags within a VARCLASS definition:

Tag	Name	Usage	Page	Required
CHECKL	Check list	Multiple	253	No
XLATL	Translate list	Multiple	495	No

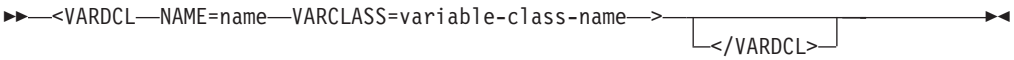
Example

The following example contains two variable classes. The first variable class provides an alphabetic validity check. The second variable class provides input translation to upper case and validates that the input is one of the listed values. Also shown in the markup are two input data fields (within a PANEL definition) that refer to the variable declarations associated with the variable classes.

```
<!DOCTYPE DM SYSTEM>


<VARCLASS NAME=namec TYPE='char 25'>
   <CHECKL>
      <CHECKI TYPE=alpha>
   </CHECKL>

<VARCLASS NAME=officec TYPE='char 4'>
   <XLATL FORMAT=upper>
   </XLATL>
   <CHECKL>
      <CHECKI TYPE=values PARM1=EQ PARM2='A101 A108 B210 B214'>
   </CHECKL>

<VARLIST>
   <VARDCL NAME=reserver VARCLASS=namec>
   <VARDCL NAME=office VARCLASS=officec>
</VARLIST>

<PANEL NAME=varclass>Conference Room
   <TOPINST>Enter the required information to reserve a conference room.
   <AREA>
      <DTACOL PMTWIDTH=20>
        <DTAFLD DATAVAR=reserver USAGE=in ENTWIDTH=25>Name
        <DTAFLD DATAVAR=office USAGE=in ENTWIDTH=4>Office number
      </DTACOL>
   </AREA>
</PANEL>
```

VARDCL (Variable Declaration)

The VARDCL tag declares variables referred to in dialog element definitions.



NAME=name
This attribute specifies the name of a variable used elsewhere in the DTL source file. The *name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

VARCLASS=variable-class-name
This attribute specifies the default variable class associated with the variable. If you want to perform a different set of checks or translations on any data field or list column, you can specify an overriding variable class in the DTAFLD or LSTCOL tags.

Description

The VARDCL tag declares variables referred to in dialog element definitions.

Note: The ISPF Dialog Tag Language conversion utility does not require that you code the VARCLASS, VARDCL, or VARLIST tags for successful generation of a panel, command table, or message member that includes variables. If the conversion utility finds a variable that does not have an associated VARDCL definition, it issues a warning message.

The use of the VARCLASS, VARDCL, and VARLIST tags is required if you want to use the facilities provided by the CHECKL and XLATL tags.

Conditions

- You must code the VARDCL tag within a VARLIST tag. See "VARLIST (Variable List)" on page 487 for a complete description of this tag.

Nested Tags

None.

Example

The following source file markup contains variable declarations for all of the variables defined in the panel definition. The declared variables include:

- The variable *whchsrch* specified in the CHECKVAR attributes associated with the pull-down choices of the **Search** action bar choice.
- The data field variables *curdate*, *cardno*, *name*, and *address*.
- The variable *cardsel*, which is the entry-field of the single-choice selection field.
- The variables *north*, *south*, *east*, and *west*, which are the entry-fields associated with the multiple-choice selection field.
- The variables defined as the check variables (CHECKVAR attribute) for the selection fields.

```
<!DOCTYPE DM SYSTEM(
  <!entity sampabc sysem>
  <!entity sampbody system>)>
```

```
<VARCLASS NAME=date    TYPE='char 8'>
<VARCLASS NAME=numcls  TYPE='numeric 7'>
<VARCLASS NAME=namecls TYPE='char 25'>
<VARCLASS NAME=char1cls TYPE='char 1'>
<VARCLASS NAME=char7cls TYPE='char 7'>

<VARLIST>
  <VARDCL NAME=whchsrch VARCLASS=char1cls>
  <VARDCL NAME=curdate VARCLASS=date>
  <VARDCL NAME=cardno  VARCLASS=numcls>
  <VARDCL NAME=name    VARCLASS=namecls>
  <VARDCL NAME=address VARCLASS=namecls>
  <VARDCL NAME=cardsel VARCLASS=char1cls>
  <VARDCL NAME=card VARCLASS=char7cls>
  <VARDCL NAME=north VARCLASS=char1cls>
  <VARDCL NAME=south VARCLASS=char1cls>
  <VARDCL NAME=east VARCLASS=char1cls>
  <VARDCL NAME=west VARCLASS=char1cls>
  <VARDCL NAME=nth VARCLASS=char1cls>
  <VARDCL NAME=sth VARCLASS=char1cls>
  <VARDCL NAME=est VARCLASS=char1cls>
  <VARDCL NAME=wst VARCLASS=char1cls>
</VARLIST>

<PANEL NAME=vardcl>Library Card Registration
<AB>
&sampabc;
</AB>
&sampbody;
</PANEL>
```

VARLIST (Variable List)

The VARLIST tag provides the means to code VARDCL tags in a single list.

Description

The VARLIST tag provides the means to code VARDCL tags in a single list. The VARDCL tags coded within a VARLIST definition declare variables that are referred to in the dialog element definitions within a DTL source file.

Note: The ISPF Dialog Tag Language conversion utility does not require that you code the VARCLASS, VARDCL, or VARLIST tags for a successful generation of a panel, command table, or message member that includes variables. If the conversion utility finds a variable that does not have an associated VARDCL definition, it issues a warning message.

The use of the VARCLASS, VARDCL, and VARLIST tags is required if you want to use the facilities provided by the CHECKL and XLATL tags.

Conditions

- The VARLIST tag requires an end tag.
- You cannot code the VARLIST tag within any other tag definition.
- You can code the VARLIST tag immediately after any and all VARCLASS tags within the DTL source file and before any tag definitions that refer to the variables declared in the variable list.

Nested Tags

You code the following tag within a VARLIST definition:

Tag	Name	Usage	Page	Required
VARDCL	Variable declaration	Multiple	487	No

Example

The following source file markup contains a variable list. The variable declarations within the list define variables for the fields within the PANEL definitions that refer to them.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=char8 TYPE='char 8'>
<VARCLASS NAME=name  TYPE='char 25'>
<VARCLASS NAME=phoncls TYPE='char 12'>
<VARCLASS NAME=appcls  TYPE='char 1'>
  <XLATL FORMAT=upper>
  </XLATL>
  <CHECKL>
    <CHECKI TYPE=values PARM1=EQ PARM2='Y N'>
  </CHECKL>

<VARLIST>
  <VARDCL NAME=curdate VARCLASS=char8>
  <VARDCL NAME=namevar VARCLASS=name>
  <VARDCL NAME=passvar VARCLASS=char8>
  <VARDCL NAME=xlname  VARCLASS=name>
  <VARDCL NAME=xphone  VARCLASS=phoncls>
  <VARDCL NAME=xapp    VARCLASS=appcls>
</VARLIST>

<PANEL NAME=varlist1 KEYLIST=keylxmp>System Log On
<TOPINST>Complete the following fields, then press Enter.
<AREA>
  <DTACOL PMTWIDTH=12>
    <DTAFLD DATAVAR=curdate ENTWIDTH=8 USAGE=out>Date
    <DTAFLD DATAVAR=namevar ENTWIDTH=25 DESWIDTH=15>Name
    <DTAFLD DATAVAR=passvar ENTWIDTH=8 DISPLAY=no>Password
  </DTACOL>
</AREA>
</PANEL>

<PANEL NAME=varlist2 DEPTH=14 KEYLIST=keyltbl>Subscriber List
<TOPINST>Enter phone number, if missing,
(format - nnn-nnn-nnnn) and approved
indicator (y or n) for each person.
<AREA>
  <LSTFLD>
      <LSTCOL DATAVAR=xlname USAGE=out COLWIDTH=25>Last Name
      <LSTCOL DATAVAR=xphone COLWIDTH=12>Phone Number
    <LSTGRP>Approved
      <LSTCOL DATAVAR=xapp USAGE=in REQUIRED=yes
        COLWIDTH=1 MSG=msgv886>(Y or N)
    </LSTGRP>
  </LSTFLD>
</AREA>
<CMDAREA>Enter a command
</PANEL>
```

VARSUB (Variable Substitution)

The VARSUB tag specifies a variable to substitute in message text.

VAR=variable-name
This attribute specifies the variable whose value is substituted within the message.

The *variable-name* should be declared with a VARDCL tag.

The *variable-name* must follow the standard naming convention described in "Rules for Variable Names" on page 211.

Description

The VARSUB tag specifies a variable to substitute in message text. You use the required VAR attribute to specify the variable whose value is resolved and inserted into the message text when the message is displayed. The value coded must be a variable name without leading % notation.

You can code the VARSUB tag in the *message-text* of a MSG tag. The variable value is inserted by ISPF at run time at the position in the message text where the VARSUB tag is coded.

For example, assume the following VARSUB tag was coded within the text of this message:

```
<msgmbr name=abca00>
<msg suffix=1 msgtype=warning>Invalid name,
"<VARSUB VAR=LASTN>", specified.
The name may contain only alphabetic characters.
</msgmbr>
```

When a dialog refers to a message *abca001* (with a GETMSG, SETMSG, DISPLAY, or TBDISPL service call) or the message is displayed by ISPF during panel validation, the value of *lastn* is retrieved and inserted into the message text. The message after substitution would appear as follows:

```
Invalid name, "Jones1", specified.
The name may contain only alphabetic characters.
```

Conditions

- You must code the VARSUB tag within the text of a MSG definition. See "MSG (Message)" on page 393 for a complete description of this tag.
- The value specified by the VAR attribute should be declared with a VARDCL tag. See "VARDCL (Variable Declaration)" on page 486 for a complete description of this tag.

Nested Tags

None.

Example

The following markup contains a message member which contains nine MSG definitions. The text of messages *MSGV883* and *MSGV888* contain variable substitutions. Figure 168 shows the generated message member.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=msgcls TYPE='char 20'>
<VARLIST>
  <VARDCL NAME=phoneno VARCLASS=msgcls>
  <VARDCL NAME=cnum    VARCLASS=msgcls>
</VARLIST>

<MSGMBR NAME=msgv88>
  <MSG SUFFIX=1>Name must be alphabetic.
  <MSG SUFFIX=2>Enter only number of days.
  <MSG SUFFIX=3 MSGTYPE=critical>The only rooms we have available
  are either SINGLE or DOUBLE.  Please call the manager of the hotel
  who will arrange equivalent lodging at another
  hotel in the area.  This is our mistake, and we will, of course,
  pick up the bill.  Please call collect <VARSUB VAR=phoneno>.
  <MSG SUFFIX=4 MSGTYPE=action LOCATION=modal>Please enter either
  BIGCHARGE, V I S T A, EZCARD, CHECK, or CASH.
  <MSG SUFFIX=5 MSGTYPE=warning LOCATION=modeless>Please enter your name.
  <MSG SUFFIX=6>Please enter Y or N.
  <MSG SUFFIX=7>Card number is a seven-digit number.
  <MSG SUFFIX=8 MSGTYPE=warning>The card number you
  entered, <VARSUB VAR=cnum> is not valid.
  <MSG SUFFIX=9>Message '9' contains imbedded quotes.
</MSGMBR>
```

```
MSGV881 .TYPE=NOTIFY
'Name must be alphabetic.'
MSGV882 .TYPE=NOTIFY
'Enter only number of days.'
MSGV883 .TYPE=CRITICAL
'The only rooms we have available are either SINGLE or DOUBLE. Please call th' +
'e manager of the hotel who will arrange equivalent lodging at another hotel ' +
'in the area. This is our mistake, and we will, of course, pick up the bill. ' +
'Please call collect &PHONENO.'
MSGV884 .TYPE=ACTION .WINDOW=RESP
'Please enter either BIGCHARGE, V I S T A, EZCARD, CHECK, or CASH.'
MSGV885 .TYPE=WARNING .WINDOW=NORESP
'Please enter your name.'
MSGV886 .TYPE=NOTIFY
'Please enter Y or N.'
MSGV887 .TYPE=NOTIFY
'Card number is a seven-digit number.'
MSGV888 .TYPE=WARNING
'The card number you entered, &CNUM is not valid.'
MSGV889 .TYPE=NOTIFY
'Message '9'' contains imbedded quotes.'
```

Figure 168. Variable Substitution

WARNING (Warning)

The WARNING tag defines text that alerts the user to a risk of possible error conditions in the system.

text
This is the text of the warning.

Description

The WARNING tag defines text that alerts the user to a risk of possible error conditions in the system.

The WARNING tag is one of the tags that alert the user of a possible risk; the others are the CAUTION tag and the ATTENTION tag.

Code a warning statement before the text to which it pertains so that the user can read about the possible risks before reading the text.

When a warning statement is displayed, the string "Warning:" (or its translated equivalent) appears on the screen before the text of the warning statement.

You can code additional paragraphs of warning text by coding the P (paragraph) tag within a WARNING definition. You can also code other tags allowed in an information area within a WARNING definition.

Conditions

- The WARNING tag requires an end tag.
- You must code the WARNING tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.
- The WARNING tag must be immediately preceded by a P, LI, or LP tag. If the WARNING tag is coded on the same line as one of these tags, there can be no intervening blanks. See "P (Paragraph)" on page 408, "LI (List Item)" on page 364, and "LP (List Part)" on page 370 for descriptions of these tags.
- You cannot nest WARNING, ATTENTION, or CAUTION tags within each other.

Nested Tags

You can code the following tags within a WARNING definition:

Tag	Name	Usage	Page	Required
DL	Definition list	Multiple	298	No
FIG	Figure	Multiple	329	No
HP	Highlighted phrase	Multiple	354	No
LINES	Lines	Multiple	366	No
NOTE	Note	Multiple	400	No
NOTEL	Note List	Multiple	402	No
NT	Note	Multiple	404	No

Tag	Name	Usage	Page	Required
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No
XMP	Example	Multiple	497	No

Example

The following help panel markup contains a warning statement. The warning statement starts at the left margin because it is imbedded in the LP tag.

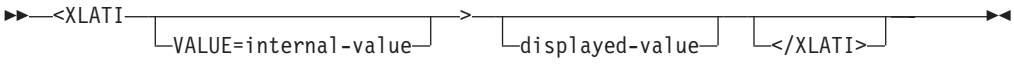
```
<!DOCTYPE DM SYSTEM>

<HELP NAME=warning DEPTH=20>Help For Changing a File
<AREA>
<INFO>
  <OL>
    <LI>Type over the existing data
    in the entry fields with the new data.
      <LP><WARNING>Performing the next step will save
      all changes and delete the existing data.
      <P>To quit this function without
      deleting the existing data, press F12.
      </WARNING>
    <LI>Press Enter to save the
    updated data.
  </OL>
</INFO>
</AREA>
</HELP>
```

```
            Help For Changing a File

1.  Type over the existing data in the entry
    fields with the new data.

Warning: Performing the next step will save all
changes and delete the existing data.

To quit this function without deleting the
existing data, press F12.

2.  Press Enter to save the updated data.




 F1=Help          F3=Exit          F5=Exhelp
 F6=Keyshelp      F7=PrvTopic      F8=NxtTopic
F10=PrvPage      F11=NxtPage      F12=Cancel
```

Figure 169. Warning Statement

XLATI (Translate Item)

The XLATI tag defines an individual list element in a translate list.



VALUE=internal-value
ISPF saves this value in the variable pool when translating on input and retrieves it from the variable pool when translating on output. If the *internal-value* contains characters other than A–Z, a–z, and 0–9, you must enclose the value in quotes.

Omitting this attribute indicates that any value is acceptable. When translating on input, ISPF does not translate the *displayed-value* before storing it in the pool. When translating on output, ISPF translates to the *displayed-value* any value that is not already matched.

displayed-value
This attribute specifies the displayed value that must be matched when doing a translation on input and the result when doing a translation on output. The test for a translation match is case-sensitive. Any characters, including embedded blanks, are allowed in the *displayed-value*. If the value has blanks that you want preserved, or the value consists of only blanks, the value should be coded within the LIT (Literal) tag. If the LIT tag is not used, all blanks will be stripped and any value with only blanks will indicate that no value was specified.

Omitting this value indicates that any value is acceptable. When translating on output, this means that the *internal-value* is not to be translated before being displayed. When translating on input, it means that any value not already matched is to translate to the *internal-value*.

Description

The XLATI tag defines an individual list element in a translate list. As many XLATI tags as are necessary (up to a limit of 126) to accomplish the desired translation can be included within the translation list.

Each XLATI tag provides information necessary to translate a *displayed-value* to an *internal-value* and vice versa. Translation is done in the order given by the tags. Translation stops when a match is found. An XLATI tag that omits both *internal-value* and *displayed-value* has the following effect: when translating on output the variable value is displayed, and when translating on input the entered value is stored in the variable.

The ISPF TRANS() function will be used for all translations. When translating on output, ISPF)INIT panel logic translates the *internal-value* to the *displayed-value*. When translating on input, ISPF)PROC panel logic translates the *displayed-value* to the *internal-value*. The test for a translation match is case-sensitive. You can code an XLATL FORMAT=UPPER definition before an XLATL definition that contains XLATI tags to convert user input to uppercase before the translate list is processed.

Conditions

You must code the XLATI tag within an XLATL definition. See "XLATI (Translate Item)" on page 493 for a complete description of this tag.

Nested Tags

You can code the following tag within an XLATI definition:

Tag	Name	Usage	Page	Required
LIT	Literal	Multiple	368	No

Example

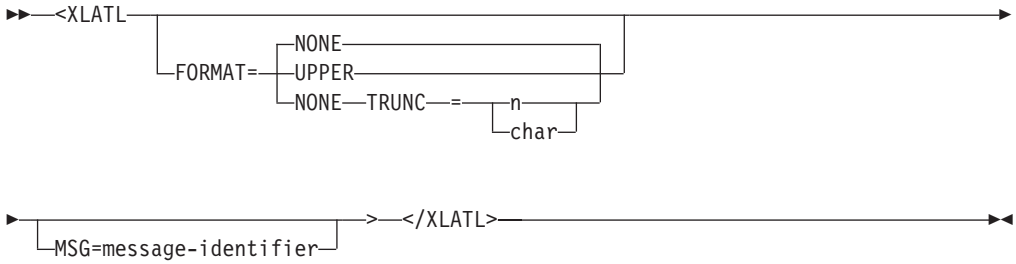
The following source file markup contains a variable class with a translate list that performs input and output translation on values assigned to the winter months. The associated variable declarations and fields are also shown.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=monthcls type='char 3'>
   <XLATL FORMAT=upper>
   </XLATL>
   <XLATL MSG=abcd003>
       <XLATI VALUE=11>NOV
       <XLATI VALUE=12>DEC
       <XLATI VALUE=01>JAN
       <XLATI VALUE=02>FEB
       <XLATI VALUE=03>MAR
   </XLATL>

<VARCLASS NAME=costcls TYPE='numeric 6' MSG=abcd001>

<VARCLASS NAME=typecls TYPE='char 4'>
   <XLATL FORMAT=upper>
   </XLATL>
   <CHECKL MSG=abcd002>
      <CHECKI TYPE=values Parm1=EQ Parm2= 'GAS OIL ELEC'>
   </CHECKL>

<VARLIST>
   <VARDCL NAME=month VARCLASS=monthcls>
```

```
   <VARDCL NAME=cost VARCLASS=costcls>
   <VARDCL NAME=heat VARCLASS=typecls>
</VARLIST>

<PANEL NAME=xlati KEYLIST=keylxmp>Heating Cost Survey
   <TOPINST>Complete the information below and then press Enter.
   <AREA>
     <DTACOL PMTWIDTH=20 DESWIDTH=30>
        <DTAFLD DATAVAR=month ENTWIDTH=3>Winter month
          <DTAFLDD>Enter Nov, Dec, Jan, Feb, or Mar
        <DTAFLD DATAVAR=cost ENTWIDTH=6>Heating cost
        <DTAFLD DATAVAR=heat ENTWIDTH=4>Type of heating
          <DTAFLDD>Enter Oil, Gas, or Elec
     </DTACOL>
   </AREA>
   <BOTINST>Thank you for your participation.
</PANEL>

<MSGMBR NAME=abcd00>
  <MSG SUFFIX=1>Heating cost must be numeric
  <MSG SUFFIX=2>Type of heating must be "Gas", "Oil", or "Elec"
  <MSG SUFFIX=3>Winter month must be "Nov","Dec", "Jan", "Feb", or "Mar"
</MSGMBR>
```

XLATL (Translate List)

The XLATL tag defines a translate list for a variable class.



FORMAT=NONE | UPPER
This attribute defines the type of translation. NONE specifies that enclosed XLATI tags are to be used to translate the value on an item for item basis. UPPER specifies that the variable value is translated to uppercase.

TRUNC=n | char
This attribute defines the type of truncation to be performed on input values. It is valid only when FORMAT=NONE. If a number is provided, truncation occurs at the length indicated. If a non-numeric character is provided, truncation occurs at the first occurrence of that character.

MSG=message-identifier
This attribute specifies the ID of a message to be issued for the error condition that results when an input translation fails because the user entered a value not specified in the list. Specifying an XLATI tag with no *internal-value* and no *displayed-value* ensures that any value not in the list is accepted without error. If no message ID is specified and an error occurs, the *message-identifier* specified on the VARCLASS tag is used. If no *message-identifier* is specified on the XLATL tag or the VARCLASS tag, no message is displayed.

Note: This message is not used if translation on output fails. The variable value is displayed as is, subject to whatever size restrictions apply to the field.

Description

The XLATL tag defines a translate list for a variable class. XLATI tags, which define the elements of the translation list, are coded within the XLATL tag. A translation list is defined within a VARCLASS tag.

If FORMAT=NONE is specified, it is expected that there are XLATI tags within the XLATL definition. If FORMAT=UPPER is specified, no XLATI tags are accepted in the XLATL definition.

Translation lists are optional and provide a means of translating between a displayed value and the internal value of the variable. Translation can occur on input (the translation result is stored in the variable pool), on output (the value from the pool is translated before the user sees it), or both, depending on the USAGE attribute of the DTAFLD tag that is associated with the variable. Translation for table display is not supported by ISPF. Refer to the *ISPF Dialog Developer's Guide and Reference* for additional information about the TRANS function.

Conditions

- The XLATL tag requires an end tag.
- You must code the XLATL tag within a VARCLASS definition. See "VARCLASS (Variable Class)" on page 482 for a complete description of this tag.
- You must code any and all XLATL tags before any CHECKL tags in the same variable class.

Nested Tags

You can code the following tag within the XLATL tag:

Tag	Name	Usage	Page	Required
XLATI	Translate item	Multiple	493	Yes

Example

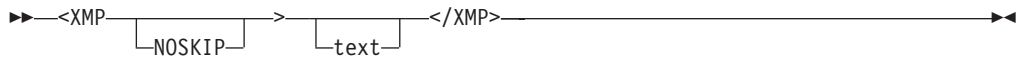
The following source file markup includes translation of user input for *monthcls* to uppercase followed by a translation list of the abbreviated month to an internal value. If no match is found, message *abcd003* is issued. The example also shows the use of uppercase translation before a check for a list of values for **Type of heating**.

```
<!DOCTYPE DM SYSTEM>

<VARCLASS NAME=monthcls type='char 3'>
   <XLATL FORMAT=upper>
   </XLATL>
   <XLATL MSG=abcd003>
       <XLATI VALUE=11>NOV
       <XLATI VALUE=12>DEC
       <XLATI VALUE=01>JAN
       <XLATI VALUE=01>FEB
       <XLATI VALUE=03>MAR
   </XLATL>

<VARCLASS NAME=costcls TYPE='numeric 6' MSG=abcd001>
```

```
<VARCLASS NAME=typecls TYPE='char 4'>
   <XLATL FORMAT=upper>
   </XLATL>
   <CHECKL MSG=abcd002>
      <CHECKI TYPE=values Parm1=EQ Parm2= 'GAS OIL ELEC'>
   </CHECKL>

<VARLIST>
   <VARDCL NAME=month VARCLASS=monthcls>
   <VARDCL NAME=cost VARCLASS=costcls>
   <VARDCL NAME=heat VARCLASS=typecls>
</VARLIST>

<PANEL NAME=xlatl KEYLIST=keylxmp>Heating Cost Survey
   <TOPINST>Complete the information below and then press Enter.
   <AREA>
     <DTACOL PMTWIDTH=20 DESWIDTH=30>
        <DTAFLD DATAVAR=month ENTWIDTH=3>Winter month
          <DTAFLDD>Enter Nov, Dec, Jan, Feb, or Mar
        <DTAFLD DATAVAR=cost ENTWIDTH=6>Heating cost
        <DTAFLD DATAVAR=heat ENTWIDTH=4>Type of heating
          <DTAFLDD>Enter Oil, Gas, or Elec
     </DTACOL>
   </AREA>
   <BOTINST>Thank you for your participation.
</PANEL>

<MSGMBR NAME=abcd00>
  <MSG SUFFIX=1>Heating cost must be numeric
  <MSG SUFFIX=2>Type of heating must be "Gas", "Oil", or "Elec"
  <MSG SUFFIX=3>Winter month must be "Nov","Dec", "Jan", "Feb", or "Mar"
</MSGMBR>
```

XMP (Example)

The XMP tag defines unformatted text within an information region.



NOSKIP
This attribute causes the blank line normally placed before the example to be skipped.

text
This is the text of the example.

Description

The XMP tag defines unformatted text within an information region.

Text coded between the XMP start and end tags is indented two spaces and formats from the current left margin. Tags which normally cause word-wrapping (for example, P and LI) will not cause word-wrapping when nested within an XMP tag.

When defining text for an example in your source file, you should be careful not to exceed the width of the information region it is coded within. If the source text on any line exceeds the formatting width, the conversion utility truncates the line. A warning message is issued the first time truncation occurs.

Conditions

- The XMP tag requires an end tag.
- You must code the XMP tag within an INFO definition. See "INFO (Information Region)" on page 356 for a complete description of this tag.
- You can code multiple XMP tags within an INFO definition, as long as they are not nested within each other.

Nested Tags

You can code the following tags within an XMP definition:

Tag	Name	Usage	Page	Required
DL	Definition list	Multiple	298	No
HP	Highlighted phrase	Multiple	354	No
NOTE	Note	Multiple	400	No
NOTEL	Note List	Multiple	402	No
NT	Note	Multiple	404	No
OL	Ordered list	Multiple	405	No
P	Paragraph	Multiple	408	No
PARML	Parameter list	Multiple	425	No
PS	Point-and-Shoot	Multiple	439	No
RP	Reference phrase	Multiple	452	No
SL	Simple list	Multiple	470	No
UL	Unordered list	Multiple	479	No

Example

The following help panel markup contains an example. Figure 170 on page 499 shows the formatted result.

```
<!DOCTYPE DM SYSTEM>

<HELP NAME=xmp WIDTH=40 DEPTH=20>Help for the Search Function
<AREA>
<INFO>
 <P>To locate a book, type the book
title in the "Title" field and press Enter.
<P>Example:
<XMP>
Title:  THE JOY OF CODING
</XMP>
<P>You don't have to worry about using
upper or lowercase letters; all text is automatically
converted to uppercase for the search.
</INFO>
</AREA>
</HELP>
```

```
     Help for the Search Function

To locate a book, type the book title
in the "Title" field and press Enter.

Example:

  Title:  THE JOY OF CODING

You don't have to worry about using
upper or lowercase letters; all text
is automatically converted to
uppercase for the search.




 F1=Help      F3=Exit      F5=Exhelp
 F6=Keyshelp  F7=PrvTopic  F8=NxtTopic
F10=PrvPage  F11=NxtPage  F12=Cancel
```

Figure 170. Example

Part 3. Appendixes

© Copyright IBM Corp. 1989, 2001

Appendix. Dialog Tag Language (DTL) Tags

Tag Summary

The following table is an alphabetic summary of the supported Dialog Tag Language (DTL) tags for z/OS V1R2.0 ISPF. The table shows the tag, tells whether an end tag is required (Yes) or optional (No), and lists the tag's attributes (if any) and the tag content (if any) in italics. The table also lists which tags you can nest within the tag, as well as which tags you can code the tag within.

Table 7. Tag summary

Tag	End tag	Attributes	Nested tags	Used within
AB	Yes	MNEMGEN=YES | NO ABSEPSTR=ab-separator-string ABSEPCHAR=ab-separator-character	ABC	PANEL
ABC	No	HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname PDCVAR=pdc-variable-name *choice-description-text*	COMMENT M PDC PDSEP SOURCE	AB
ACTION	No	RUN=internal-command-name | %varname PARM=parameters | %varname APPLCMD=NO | YES TYPE=CMD | PGM | PANEL | WSCMD | WSCMDV | EXIT NEWAPPL | NEWAPPL=application-id NEWWINDOW PASSLIB NEWPOOL SUSPEND SCRNAME=screen-name NOCHECK ADDPOP OPT=option | %varname MODE=LINE | FSCR LANG=APL | CREX BARRIER NEST WSDIR=ws-directory WSINVOKE=MODELESS | MODAL WSSIZE=MAX | MIN WSVIEW=VIS | INVIS SETVAR=variable-name VALUE=1 | string | %varname TOGVAR=variable-name VALUE1=0 | string | %varname VALUE2=1 | string | %varname		CHOICE PDC

© Copyright IBM Corp. 1989, 2001

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
AREA	Yes	MARGINW=1 | n MARGIND=0 INDENT=n DEPTH=n | * EXTEND=OFF | ON | FORCE DIV=NONE | BLANK | SOLID | DASH | TEXT DIVWIDTH=MAX | MIN FORMAT=START | CENTER | END TEXT=divider-text WIDTH=n DIR=VERT | HORIZ	COMMENT DA DIVIDER DTACOL DTAFLD GA GENERATE GRPHDR INFO LSTFLD PNLINST REGION SELFLD SOURCE	HELP PANEL
ASSIGNI	No	VALUE=test-value RESULT=assigned-value		ASSIGNL
ASSIGNL	Yes	DESTVAR=destination-variable-name	ASSIGNI	DTAFLD
ATTENTION	Yes	*text*	DL FIG HP LINES NOTE NOTEL NT OL P PARML PS RP SL UL XMP	LI LP P

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
ATTR	No	ATTRCHAR=code TYPE=DATAIN | DATAOUT | CHAR INTENS=HIGH | LOW | NON | %varname CAPS=OFF | ON | IN | OUT | %varname JUST=ASIS | LEFT | RIGHT | %varname PAD=NULLS | USER | char | %varname PADC=NULLS | USER | char | %varname SKIP=OFF | ON | %varname GE=OFF | ON | %varname COLOR=WHITE | RED | BLUE | GREEN | PINK | YELLOW | TURQ | %varname HILITE=USCORE | BLINK | REVERSE | %varname NUMERIC=OFF | ON | %varname FORMAT=EBCDIC | DBCS | MIX | %varname OUTLINE=NONE | L | R | O | U | BOX | %varname PAS=OFF | ON | %varname CKBOX=OFF | ON | %varname CUADYN=CEF | EE | LEF | NEF | VOI | LID | LI | CH | CT | DT | ET | FP | NT | PIN | PT | SAC | SI | SUC | WASL | WT | %varname CSRGRP=NO | YES | n ATTN=OFF | ON | %varname		DA
BOTINST	No	COMPACT *instruction-text*	HP PS RP	PANEL
CAUTION	Yes	*text*	DL FIG HP LINES NOTE NOTEL NT OL P PARML PS RP SL UL XMP	LI LP P
CHDIV	No	TYPE=NONE | SOLID | DASH | TEXT GUTTER=1 | n FORMAT=START | CENTER | END *divider-text*	HP	SELFLD CHOICE

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
CHECKI	No	TYPE= RANGE PARM1=low-bound | %varname PARM2=high-bound | %varname ALPHA CHARS PARM1=EQ PARM2=character-set VALUES PARM1=EQ PARM2=value-list VALUESX PARM1=NE PARM2=value-list BIT NAME NAMEF PICT PARM1=EQ PARM2=pictstring PICTCN PARM1=mask-character PARM2=field-mask PARM3=string NUM DBCS LISTV PARM1=EQ PARM2=%varlist LISTVX PARM1=NE PARM2=%varlist ALPHAB LEN PARM1=operator | %varname PARM2=length | %varname EBCDIC ENUM DSNAME DSNAMEF DSNAMEFM DSNAMEPQ DSNAMEQ MIX HEX FILEID INCLUDE PARM1=IMBLK PARM2=ALPHA | ALPHAB | NUM PARM3=ALPHA | ALPHAB | NUM IDATE STDDATE JDATE JSTD ITIME STDTIME IPADDR4		CHECKL

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
CHECKL	Yes	MSG=message-identifier	CHECKI	VARCLASS
CHOFLD	No	DATAVAR=field-data VARCLASS=variable-class-name HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname USAGE=BOTH | IN | OUT REQUIRED=NO | YES MSG=message-identifier AUTOTAB=NO | YES ENTWIDTH=n FLDSPACE=n ALIGN=START | CENTER | END DISPLAY=YES | NO NOENDATTR PAD=NULLS | USER | char | %varname PADC=NULLS | USER | char | %varname OUTLINE=NONE | L | R | O | U | BOX | %varname PSVAR=point-and-shoot-variable | %varname PSVAL=point-and-shoot-value | %varname PAS=%varname EXPAND ATTRCHANGE=NO | YES | NEW INIT=initial-value IMAPNAME=image-name | %varname IMAPNAMEP=image-namep | %varname PLACE=ABOVE | BELOW | LEFT | RIGHT | %varname ATTRCHAR=code CAPS=OFF | ON *choice-description-text*	ACTION COMMENT HP PS RP SOURCE	CHOICE
CHOICE	No	NAME=choice-name HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname CHECKVAR=variable-name MATCH=1 | string NOMATCH=0 | string AUTOTAB=YES | NO SELCHAR='char(s),n' PAD=NULLS | USER | char | %varname PADC=NULLS | USER | char | %varname OUTLINE=NONE | L | R | O | U | BOX | %varname HIDE HIDEX UNAVAIL=variable-name UNAVAILMAT=1 | string TRUNC=n AUTOSEL=YES | NO *choice-description-text*	ACTION CHOFLD COMMENT HP PS RP SOURCE	SELFLD
CMD	No	NAME=internal-command-name ALTDESCR=command-description *external-command-name*	CMDACT T	CMDTBL

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
CMDACT	No	ACTION= 'SELECT=select-parameters' 'ALIAS=internal-command-name parameters' PASSTHRU SETVERB BACKWARD CANCEL EXIT EXHELP FKA FORWARD HELP PANELID RETRIEVE %varname application-command ASIS		CMD
CMDAREA	No	HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname PMTLOC=BEFORE NOINIT PAD=NULLS | USER | char | %varname PADC=NULLS | USER | char | %varname OUTLINE=NONE | L | R | O | U | BOX | %varname NAME=cmdarea-variable-name ENTWIDTH=n PMTTEXT=YES | NO CMDLOC=DEFAULT | ASIS CMDLEN=DEFAULT | MAX AUTOTAB=NO | YES SCROLLVAR=scroll-variable SCRVHELP=NO | YES | scroll-help-panel-name |*scroll-help-message-id | %varname | *%varname SCROLLTAB=NO | YES SCRCAPS=OFF | ON PSBUTTON=cmd-pb-text PSVAR=point-and-shoot-variable | %varname PSVAL=point-and-shoot-value | %varname IMAPNAME=image-name | %varname IMAPNAMEP=image-namep | %varname PLACE=ABOVE | BELOW | LEFT | RIGHT | %varname CAPS=OFF | ON NOJUMP=OFF | ON VARDCL=YES | NO *command-prompt-text*	HP	PANEL
CMDTBL	Yes	APPLID=application-identifier SORT=NO | YES	CMD	

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
COMMENT	No	TYPE=END | CCSID | PANEL | ATTR | ABCINIT | ABCPROC | INIT | REINIT | PROC | HELP | PNTS | LIST *comment-text*		ABC AREA CHOICE DA DTACOL DTAFLD HELP LSTCOL LSTFLD LSTGRP MSGMBR PANEL PDC REGION SELFLD
COMPOPT	No	REPLACE | NOREPLACE SCREEN | DISK NODBCS | DBCS NOKANA | KANA KEYLAPPL=xxxx NOPANEL | PANEL NOMSGSUPP | MSGSUPP NOCUASUPP | CUASUPP PREP | NOPREP CUAATTR | NOCUAATTR NOLSTVIEW | LSTVIEW STATS | NOSTATS NOSCRIPT | SCRIPT NOLISTING | LISTING NOFORMAT | FORMAT NOMSGEXPAND | MSGEXPAND LOGREPL | NOLOGREPL LISTREPL | NOLISTREPL ACTBAR | NOACTBAR GUI | NOGUI VERSION | NOVERSION NOMERGESAREA | MERGESAREA NODISPLAY | DISPLAY NODISPLAYW | DISPLAYW DSNCHK | NODSNCHK GRAPHIC | NOGRAPHIC ZVARS | NOZVARS NODBALIGN | DBALIGN NOMCOMMENT | MCOMMENT NOVPADC | PADC ADD RESET *national-language*	None	
COPYR	No	*copyright-text*		

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
DA	Yes	NAME=varname EXTEND=OFF | ON | FORCE LVLINE=variable-name SCROLL=OFF | ON | CMDLINE USERMOD=usermod-code | %varname DATAMOD=datamod-code | %varname DEPTH=n | * WIDTH=n SHADOW=shadow-name DIV=NONE | BLANK | SOLID | DASH | TEXT FORMAT=START | CENTER | END TEXT=divider-text SCROLLVAR=scroll-variable SCRVHELP=NO | YES | scroll-help-panel-name |*scroll-help-message-id | %varname | *%varname SCROLLTAB=NO | YES SCRCAPS=OFF | ON INITATTR=NT | CT | ET | WT | WASL HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname	ATTR COMMENT SOURCE	AREA PANEL REGION
DD	No	*definition-description*	DL FIG HP LINES NOTE NOTEL NT OL P PARML PS RP SL UL XMP	DL
DDHD	No	*definition-description-header*	HP PS RP	DL
DIVIDER	No	TYPE=NONE | SOLID | DASH | TEXT GAP=YES | NO GUTTER=1 | n NOENDATTR FORMAT=START | CENTER | END *divider-text*	HP	AREA DTACOL PANEL REGION

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
DL	Yes	TSIZE=10 | 'S1, S2,... Sn' BREAK=NONE | FIT | ALL COMPACT NOSKIP INDENT=n FORMAT=START | CENTER | END DIVEND=NO | YES	DD DDHD DLDIV DT DTHD DTDIV DTHDIV	ATTENTION CAUTION DD FIG INFO LI LINES LP NT PD WARNING XMP
DLDIV	No	TYPE=NONE | SOLID | DASH | TEXT GAP=YES | NO GUTTER=1 | n FORMAT=START | CENTER | END *divider-text*	HP	DL
DT	No	FORMAT=START | CENTER | END NOSKIP *definition-term*	HP PS RP	DL
DTACOL	Yes	PMTWIDTH=n | * | ** ENTWIDTH=n DESWIDTH=n | * SELWIDTH=n | * FLDSPACE=n PAD=NULLS | USER | char | %varname PADC=NULLS | USER | char | %varname OUTLINE=NONE | L | R | O | U | BOX | %varname PMTFMT=CUA | ISPF | NONE | END AUTOTAB=NO | YES ATTRCHANGE=NO | YES | NEW PMTLOC=BEFORE | ABOVE DBALIGN=YES | NO | PROMPT | FIELD | FORCE VARCLASS=variable-class-name REQUIRED=NO | YES CAPS=OFF | ON VARDCL=YES | NO	COMMENT DIVIDER DTAFLD GRPHDR SELFLD SOURCE	AREA PANEL REGION

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
DTAFLD	No	NAME=field-name DATAVAR=field-data VARCLASS=variable-class-name HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname USAGE=BOTH | IN | OUT REQUIRED=NO | YES MSG=message-identifier AUTOTAB=NO | YES ENTWIDTH=n PMTWIDTH=n | * | ** DESWIDTH=n | * FLDSPACE=n ALIGN=START | CENTER | END PMTLOC=BEFORE | ABOVE DISPLAY=YES | NO NOENDATTR PAD=NULLS | USER | char | %varname PADC=NULLS | USER | char | %varname OUTLINE=NONE | L | R | O | U | BOX | %varname PMTFMT=CUA | ISPF | NONE | END PSVAR=point-and-shoot-variable | %varname PSVAL=point-and-shoot-value | %varname PAS=%varname CSRGRP=NO | YES | n EXPAND FLDWIDTH=n ATTRCHANGE=NO | YES | NEW INIT=initial-value DEPTH=n | %varname IMAPNAME=image-name | %varname IMAPNAMEP=image-namep | %varname PLACE=ABOVE | BELOW | LEFT | RIGHT | %varname DBALIGN=YES | NO | PROMPT | FIELD | FORCE PMTSKIP=NO | YES DESSKIP=NO | YES FLDTYPE=CUA | ISPF COLOR=WHITE | RED | BLUE | GREEN | PINK | YELLOW | TURQ | %varname INTENS=HIGH | LOW | NON | %varname HILITE=USCORE | BLINK | REVERSE | %varname ATTRCHAR=code CAPS=OFF | ON NOJUMP=OFF | ON AUTOTYPE=PROJECT | GROUP1 | GROUP2 | GROUP3 | GROUP4 | TYPE | MEMBER | DSN AUTOVOL=volser-name AUTODMEM=YES | NO VARDCL=YES | NO *prompt-text*	ASSIGNL COMMENT DTAFLDD HP PS RP SOURCE	AREA DTACOL PANEL REGION

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
DTAFLDD	No	*description*	HP PS RP	DTAFLD
DTDIV	No			DL
DTHD	No	*definition-term-header*	HP PS RP	DL
DTHDIV	No			DL
FIG	Yes	FRAME=RULE | NONE WIDTH=PAGE | COL NOSKIP *figure-content*	DL FIGCAP HP NOTE NOTEL NT OL P PARML PS RP SL UL XMP	ATTENTION CAUTION DD INFO LI LP NT PD WARNING
FIGCAP	No	*figure-caption-text*	HP PS RP	FIG
GA	No	NAME=graphic-area-name EXTEND=OFF | ON | FORCE DEPTH=n | * WIDTH=n DIV=NONE | BLANK | SOLID | DASH | TEXT FORMAT=START | CENTER | END TEXT=divider-text LVLINE=variable-name		AREA PANEL REGION
GENERATE	Yes		ATTR COMMENT SOURCE	AREA HELP PANEL REGION
GRPHDR	No	FORMAT=START | CENTER | END | NONE WIDTH=n FMTWIDTH=n INDENT=n HEADLINE=NO | YES DIV=NONE | BLANK | SOLID | DASH DIVLOC=AFTER | BEFORE | BOTH COMPACT STRIP *group-heading-text*	HP PS RP	AREA DTACOL PANEL REGION

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
HELP	Yes	NAME=help-panel-name HELP=hhelp-panel-name | %varname HELPDEF=helpdef-id WIDTH=50 | n | FIT DEPTH=10 | n | FIT CCSID=n TUTOR KEYLIST=key-list-name KEYLTYPE=PRIVATE | SHARED APPLID=application-id EXPAND=xy WINTITLE=window-title APPTITLE=application-title MERGESAREA=NO | YES MSGLINE=YES | NO IMAPNAME=image-name | %varname IMAPROW=n | %varname IMAPCOL=n | %varname ZUP=zup-id ZCONT=zcont-id *help-panel-title*	AREA COMMENT DIVIDER GENERATE HP INFO REGION SOURCE TEXTLINE	
HELPDEF	No	ID=helpdef-id HELP=hhelp-panel-name | %varname WIDTH=n | FIT DEPTH=n | FIT CCSID=n KEYLIST=key-list-name KEYLTYPE=PRIVATE | SHARED APPLID=application-id EXPAND=xy WINTITLE=window-title APPTITLE=application-title MERGESAREA=NO | YES IMAPNAME=image-name | %varname IMAPROW=n | %varname IMAPCOL=n | %varname		
H1	No	COMPACT *heading-text*		INFO
H2/H3/H4	No	COMPACT *heading-text*	HP PS RP	INFO

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
HP	Yes	TYPE=ET | CH | CT | FP | LEF | LI | NT | PT | SAC | TEXT | WASL | WT COLOR=WHITE | RED | BLUE | GREEN | PINK | YELLOW | TURQ | %varname INTENS=HIGH | LOW | NON | %varname HILITE=USCORE | BLINK | REVERSE | %varname INTENSE=varname *phrase-to-be-highlighted*		ATTENTION BOTINST CAUTION CHDIV CHOICE CMDAREA DD DDHD DIVIDER DT DTAFLD DTAFLDD DTHD FIG FIGCAP GRPHDR H2 H3 H4 HELP LI LINES LP LSTCOL LSTGRP NOTE NT P PANEL PD PNLINST PT SELFLD TOPINST WARNING XMP
INFO	Yes	WIDTH=format-width | * INDENT=n	DIVIDER DL FIG Hn LINES NOTE NOTEL NT OL P PARML SL SOURCE UL XMP	AREA HELP PANEL REGION

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
KEYI	No	KEY=virtual-key CMD=internal-command-name CASE=UPPER | MIXED FKA=NO | YES | LONG | SHORT PARM=parm-string *FKA-text*		KEYL
KEYL	Yes	NAME=key-list-name HELP=help-panel-name ACTION=UPDATE | DELETE APPLID=application-id	KEYI	
LI	No	SPACE=NO | YES NOSKIP *item-text*	ATTENTION CAUTION DL FIG HP LINES NOTE NOTEL NT OL P PARML PS RP SL UL WARNING XMP	NOTEL OL SL UL
LINES	Yes	NOSKIP *text*	DL HP NOTE NOTEL NT OL P PARML PS RP SL UL XMP	ATTENTION CAUTION DD INFO LI LP NT PD WARNING
LIT	Yes	*literal-display-value*		XLATI

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
LP	No	NOSKIP *implied-paragraph*	ATTENTION CAUTION DL FIG HP LINES NOTE NOTEL NT OL P PARML PS RP SL UL WARNING XMP	NOTEL OL SL UL
LSTCOL	No	DATAVAR=column-data VARCLASS=variable-class-name HELP=NO | YES | help-panel-name | * help-message-id | %varname | *%varname USAGE=BOTH | IN | OUT REQUIRED=NO | YES MSG=message-id COLWIDTH=data-width ALIGN=START | CENTER | END AUTOTAB=NO | YES LINE=n CLEAR POSITION=n FORMAT=START | CENTER | END TEXT=descriptive-text TEXTLOC=BEFORE | AFTER TEXTFMT=START | CENTER | END TEXTLEN=n TEXTSKIP=NO | YES NOENDATTR PAD=NULLS | USER | char | %varname PADC=NULLS | USER | char | %varname OUTLINE=NONE | L | R | O | U | BOX | %varname PAS=OFF | ON | %varname CSRGRP=NO | YES | n ATTRCHANGE=NO | YES | NEW COLSPACE=n COLTYPE=CUA | ISPF | EE | VOI | LID COLOR=WHITE | RED | BLUE | GREEN | PINK | YELLOW | TURQ | %varname INTENS=HIGH | LOW | NON | %varname HILITE=USCORE | BLINK | REVERSE | %varname CAPS=OFF | ON DISPLAY=YES | NO VARDCL=YES | NO *column-heading*	COMMENT HP PS RP SOURCE	LSTFLD LSTGRP

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
LSTFLD	Yes	RULES=NONE | HORIZ | VERT | BOTH ROWS=NOSCAN | SCAN | %varname DIV=NONE | BLANK | SOLID | DASH | char SCROLLVAR=scoll-variable SCRVHELP=NO | YES | scroll-help-panel-name |*scroll-help-message-id | %varname | *%varname SCROLLTAB=NO | YES SCRCAPS=OFF | ON ATTRCHANGE=NO | YES | NEW VARDCL=YES | NO	COMMENT LSTCOL LSTGRP LSTVAR SOURCE	AREA PANEL REGION
LSTGRP	Yes	HEADLINE=NO | YES | DASH ALIGN=CENTER | START | END *column-group-heading*	COMMENT HP LSTCOL LSTGRP LSTVAR PS RP SOURCE	LSTFLD LSTGRP
LSTVAR	No	DATAVAR=variable-model-name LINE=n *column-heading*	COMMENT HP PS RP SOURCE	LSTFLD LSTGRP
M	No	*mnemonic-character*		ABC PDC
MSG	No	SUFFIX=message-suffix-number HELP=help-panel-name | %varname | * MSGTYPE=INFO | WARNING | ACTION | CRITICAL | %varname LOCATION=AREA | MODAL | MODAL(L) | MODELESS | MODELESS (L) | %varname DISP=KANA | NOKANA ALARM=NO | YES | %varname ABBREV=NONE | KEYWORD | VALUE | BOTH FORMAT=FLOW | ASIS SMSG=short-message-text *message-text*	VARSUB	MSGMBR
MSGMBR	Yes	NAME=message-member-name CCSID=n WIDTH=76 | 68	COMMENT MSG	
NOTE	No	NOSKIP TEXT=alternate-note-heading *note-text*	HP PS RP	ATTENTION CAUTION DD FIG INFO LI LINES LP PD WARNING XMP

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
NOTEL	Yes	COMPACT NOSKIP SPACE=NO | YES INDENT=n TEXT=alternate-note-heading	LI LP	ATTENTION CAUTION DD FIG INFO LI LINES LP PD WARNING XMP
NT	Yes	NOSKIP TEXT=alternate-note-heading *note-text*	DL FIG HP LINES OL P PARML PS RP SL UL XMP	ATTENTION CAUTION DD FIG INFO LI LINES LP PD WARNING XMP
OL	Yes	COMPACT NOSKIP SPACE=NO | YES INDENT=n TEXT=OL-heading-text	LI LP	ATTENTION CAUTION DD FIG INFO LI LINES LP NT PD WARNING XMP
P	No	COMPACT INTENSE=varname INDENT=n *paragraph-text*	ATTENTION CAUTION HP PS RP WARNING	ATTENTION CAUTION DD FIG INFO LI LINES LP NT PD WARNING XMP

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
PANDEF	No	ID=pandef-id HELP=help-panel-name | %varname DEPTH=n | FIT WIDTH=n | FIT | %varname KEYLIST=key-list-name KEYLTYPE=PRIVATE | SHARED APPLID=application-id CCSID=n WINDOW=YES | NO WINTITLE=window-title APPTITLE=application-title PAD=NULLS | USER | char | %varname PADC=NULLS | USER | char | %varname OUTLINE=NONE | L | R | O | U | BOX | %varname EXPAND=xy MERGESAREA=NO | YES ENTKEYTEXT=enter-key-text IMAPNAME=image-name | %varname IMAPROW=n | %varname IMAPCOL=n | %varname TMARGIN=n BMARGIN=n		

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
PANEL	Yes	NAME=panel-name HELP=help-panel-name | %varname PANDEF=pandef-id DEPTH=22 | n | FIT WIDTH=76 | n | FIT | %varname KEYLIST=key-list-name KEYLTYPE=PRIVATE | SHARED APPLID=application-id CURSOR=cursor-field CSRINDEX=index-value CSRPOS=position-value CCSID=n MENU PRIME TUTOR WINDOW=YES | NO WINTITLE=window-title APPTITLE=application-title PAD=NULLS | USER | char | %varname PADC=NULLS | USER | char | %varname OUTLINE=NONE | L | R | O | U | BOX | %varname EXPAND=xy MSGLINE=YES | NO TITLINE=YES | NO CMDLINE=YES | NO ATTRUSE=NO | YES | ALL ENDATTR=DEFAULT | TEXT TYPE=BOTH | GUI | NOGUI SMSG=short-msg-fieldname LMSG=long-msg-fieldname ASIS ACTBAR MERGESAREA=NO | YES PANELSTMT=YES | NO ENTKEYTEXT=enter-key-text IMAPNAME=image-name | %varname IMAPROW=n | %varname IMAPCOL=n | %varname TMARGIN=n BMARGIN=n ERRORCHECK=NO | YES ZUP=zup-id ZCONT=zcont-id AUTONRET=NO | YES AUTOTCMD=NO | YES *panel-title-text*	AB AREA BOTINST CMDAREA COMMENT DA DIVIDER DTACOL DTAFLD GA GENERATE GRPHDR HP INFO LSTFLD PNLINST REGION SELFLD SOURCE TEXTLINE TOPINST	

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
PARML	Yes	TSIZE=10 | 'S1 S2... Sn' BREAK=ALL | FIT | NONE COMPACT SKIP INDENT=n FORMAT=START | CENTER | END DIVEND=NO | YES	PLDIV PT PTDIV PD	ATTENTION CAUTION DD FIG INFO LI LINES LP NT PD WARNING XMP
PD	No	*parameter-description*	DL FIG HP LINES NOTE NOTEL NT OL P PARML PS RP SL UL XMP	PARML
PDC	No	HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname UNAVAIL=unavail-variable-name CHECKVAR=check-variable-name MATCH=1 | match-string ACC1=key1 ACC2=key2 ACC3=key3 *pull-down-description-text*	ACTION COMMENT M SOURCE	ABC
PDSEP	No			PDC
PLDIV	No	TYPE=NONE | SOLID | DASH | TEXT GAP=YES | NO GUTTER=1 | n FORMAT=START | CENTER | END *divider-text*	HP	PARML
PNLINST	No	COMPACT *instruction-text*	HP PS RP	AREA REGION PANEL

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
PS	Yes	VAR=point-and-shoot-variable-name | %varname VALUE=point-and-shoot-value | %varname | * CSRGRP=NO | YES | n DEPTH=n | %varname IMAPNAME=image-name | %varname IMAPNAMEP=image-namep | %varname PLACE=ABOVE | BELOW | LEFT | RIGHT | %varname *point-and-shoot-text*		ATTENTION BOTINST CAUTION CHOFLD CHOICE DD DDHD DT DTAFLD DTAFLDD DTHD FIG FIGCAP GRPHDR H2 H3 H4 LI LINES LP LSTCOL LSTGRP NOTE NT P PD PNLINST PT SELFLD TOPINST WARNING XMP
PT	No	FORMAT=START | CENTER | END NOSKIP *parameter-term*	HP PS RP	PARML
PTDIV	No			PARML
REGION	Yes	DIR=VERT | HORIZ INDENT=n WIDTH=n | * DEPTH=n | * EXTEND=OFF | ON | FORCE ALIGN=YES | NO GRPBOX=NO | YES GRPWIDTH=n GRPBXVAR=variable-name GRPBXMAT=1 | string LOCATION=DEFAULT | TITLE *group-box-title*	COMMENT DA DIVIDER DTACOL DTAFLD GA GENERATE GRPHDR INFO LSTFLD PNLINST REGION SELFLD SOURCE	AREA HELP PANEL REGION

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
RP	Yes	HELP= help-panel-name | help-message-id | %varname | *%varname *reference-phrase*		ATTENTION BOTINST CAUTION CHOFLD CHOICE DD DDHD DT DTAFLD DTAFLDD DTHD FIG FIGCAP GRPHDR H2 H3 H4 LI LINES LP LSTCOL LSTGRP NOTE NT P PD PNLINST PT SELFLD TOPINST WARNING XMP

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
SELFLD	Yes	NAME=field-name HELP=NO | YES | help-panel-name | *help-message-id | %varname | *%varname TYPE=SINGLE | MULTI | MENU | MODEL | TUTOR PMTLOC=ABOVE | BEFORE PMTWIDTH=n | * | ** SELWIDTH=n | * ENTWIDTH=2 | n | 'e1 e2...en' REQUIRED=NO | YES MSG=message-identifier FCHOICE=1 | 0 AUTOTAB=YES | NO DEPTH=n | * EXTEND=OFF | ON | FORCE TRAIL='trail-var-1 trail-var-2 ... trail-var-n' CHOICECOLS=1 | n CHOICEDEPTH=n | * CWIDTHS='w1 w2...wn' PAD=NULLS | USER | char | %varname PADC=NULLS | USER | char | %varname OUTLINE=NONE | L | R | O | U | BOX | %varname SELMSG=selfld-msg-identifier SELMSGU=selfld-msg-unavailable INIT=YES | NO | init-value VERIFY=YES | NO REFRESH=YES | NO SELFMT=START | END CHKBOX=YES | NO ZGUI=YES | NO CSRGRP=NO | YES | n TSIZE='s1 s2...sn' LISTTYPE=RADIO | LISTBOX | DDLIST | COMBO LISTREF=list-name LISTDEPTH=n DBALIGN=YES | NO | FIELD | FORCE NOSEL=no-selection-value SELDEFAULT=x PMTSKIP=NO | YES FLDTYPE=CUA | ISPF COLOR=WHITE | RED | BLUE | GREEN | PINK | YELLOW | TURQ | %varname INTENS=HIGH | LOW | NON | %varname HILITE=USCORE | BLINK | REVERSE | %varname SELCHECK=NO |YES VARDCL=YES | NO *field-prompt-text*	CHDIV CHOICE COMMENT HP PS RP SOURCE	AREA DTACOL PANEL REGION

Summary of DTL Tags

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
SL	Yes	COMPACT NOSKIP SPACE=NO | YES INDENT=n TEXT='SL-heading-text'	LI LP	ATTENTION CAUTION DD FIG INFO LI LINES LP NT PD WARNING XMP
SOURCE	Yes	TYPE=PROC | REINIT | INIT | ABCINIT | ABCPROC *text*		ABC AREA CHOICE DA DTACOL DTAFLD HELP LSTCOL LSTFLD LSTGRP PANEL PDC REGION SELFLD
T	No			CMD
TEXTLINE	Yes		DTAFLD TEXTSEG	HELP PANEL
TEXTSEG	No	EXPAND=AFTER | BEFORE | BOTH WIDTH=n *text*	HP	TEXTLINE
TOPINST	No	COMPACT *instruction-text*	HP PS RP	PANEL
UL	Yes	COMPACT NOSKIP SPACE=NO | YES INDENT=n TEXT=UL-heading-text	LI LP	ATTENTION CAUTION DD FIG INFO LI LINES LP NT PD WARNING XMP

Table 7. Tag summary (continued)

Tag	End tag	Attributes	Nested tags	Used within
VARCLASS	No	NAME=variable-class-name TYPE='CHAR maximum length' 'DBCS maximum length' 'MIXED maximum length' 'ANY maximum length' 'EBCDIC maximum length' '%varname maximum length' ITIME STDTIME IDATE STDDATE JDATE JSTD 'VMASK maximum-length' 'NUMERIC total-digits 0 | fractional-digits' MSG=message-identifier	CHECKL XLATL	
VARDCL	No	NAME=name VARCLASS=variable-class-name		VARLIST
VARLIST	Yes		VARDCL	
VARSUB	No	VAR=variable-name		MSG
WARNING	Yes	*text*	DL FIG HP LINES NOTE NOTEL NT OL P PARML PS RP SL UL XMP	LI LP P
XLATI	No	VALUE=internal-value *displayed-value*	LIT	XLATL
XLATL	Yes	FORMAT=NONE | UPPER TRUNC=n | char MSG=message-identifier	XLATI	VARCLASS
XMP	Yes	NOSKIP *text*	DL HP NOTE NOTEL NT OL P PARML PS RP SL UL	ATTENTION CAUTION DD FIG INFO LI LINES LP NT PD WARNING

Summary of DTL Tags

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non_IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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, USA.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries in writing to

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corp. 1989, 2001

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 the IBM Corporation, Department TL3B, 3039 Cornwallis Road, Research Triangle Park, North Carolina, 27709–2195, USA. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non_IBM products should be addressed to the suppliers of those products.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This book primarily documents information that is NOT intended to be used as Programming Interfaces of ISPF.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of ISPF. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

```
-----------Programming Interface information----------------
```

Trademarks

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

BookManager
C++
DFSMSdfp
DFSMSdss
DFSMShsm
DFSMSrmm
DFSMS/MVS
DFSORT
ESCON
FFST
GDDM
IBM
Language Environment
MVS
MVS/ESA
OS/2
OS/390
OS/390 Security Server
RACF
Resource Access Control Facility
SOMobjects
System View
VisualLift
VTAM

Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Index

Special Characters

% keyword
  on ENTITY statement 202
% notation 12

A

AB (action bar) tag 39
  conditions of usage 212
  defining an action bar 36
  description 212
  examples
    adding HELP attribute 39
    application panel markup 213
    defining help panel 39
    markup 38
    using mnemonic selection 40
  how to code 38
  syntax 211
ABBREV attribute
  on MSG tag 395
ABC (action bar choice) tag
  conditions of usage 215
  description 215
  examples
    defining help panel 39
    markup 38
    PDCVAR attribute 215
    using mnemonic selection 40
  syntax 214
ABSEPCHAR attribute on AB tag 212
ABSEPSTR attribute on AB tag 212
ACC1 attribute on PDC tag 431
ACC2 attribute on PDC tag 431
ACC3 attribute on PDC tag 431
accelerator attributes on PDC tag 431
ACTBAR attribute
  on PANEL tag 421
ACTBAR option 187
ACTION (action) tag
  conditions of usage 222
  description 221
  examples
    defining help panel 39
    markup 38
    PDC tags and SETVAR attribute 222
    using mnemonic selection 40
  for pull-down choice 38
  syntax 217
ACTION attribute
  on CMDACT tag 271
  on KEYL tag 362
  value ALIAS on CMDACT tag 271
  value application-command on CMDACT tag 272
  value BACKWARD on CMDACT tag 271
  value CANCEL on CMDACT tag 271
ACTION attribute *(continued)*
  value EXHELP on CMDACT tag 272
  value EXIT on CMDACT tag 272
  value FKA on CMDACT tag 272
  value FORWARD on CMDACT tag 272
  value HELP on CMDACT tag 272
  value PANELID on CMDACT tag 272
  value PASSTHRU on CMDACT tag 271
  value RETRIEVE on CMDACT tag 272
  value SELECT on CMDACT tag 271
  value SETVERB on CMDACT tag 271
action bar 39
  defining 36
  description 6
  providing help for 39
action bar (AB) tag
  conditions of usage 212
  description 212
  examples 213
  syntax 211
action bar choice 39
action bar choice (ABC) tag
  conditions of usage 215
  description 215
  examples 215
  syntax 214
action message
  defining 161
  description 161
ADDPOP attribute on ACTION tag 220
ALARM attribute
  on MSG tag 395
alerting users
  using the ATTENTION tag 139
  using the CAUTION tag 139
  using the NT tag 139
  using the WARNING tag 139
ALIAS command action
  defining 167
  description 167
ALIGN attribute
  on CHOFLD tag 257
  on DTAFLD tag 317
  on LSTCOL tag 375
  on LSTGRP tag 387
  on REGION tag 447
  using 90
ALPHA test
  description 71
  example 71
  specifying on CHECKI tag 246
ALPHAB test
  description 76
  example 76
  specifying on CHECKI tag 250
alphabetic test
  description 71
  example 71
ALTDESCR attribute on CMD tag 269
APPLCMD attribute on ACTION tag 219
application command table 165
  defining commands 165
  defining with CMDTBL tag 165
  overview 9
application-id
  ISPx 165
  option 185
  using with CMDTBL tag 165
application panel
  action bar 6
  bottom instruction 6
  command area 7
  defining a command area 55
  defining a panel ID 30
  defining a region 51
  defining action bar and pull-downs 36
  defining an area divider 50
  defining common attributes and values 57
  defining common elements 29
  defining cursor placement 32
  defining panel width and depth 31
  defining the panel title 31
  defining top and bottom instructions 42
  defining with the PANEL tag 29
  description of PANEL tag 423
  examples of PANEL tag 424
  function key area 7
  layout 5
  overview 5
  panel body 6
  panel title 6
  providing help for 32
  specifying a key mapping list 31
  syntax for PANEL tag 415
  top instruction 6
APPLID attribute
  description 165
  on CMDTBL tag 280
  on HELP TAG 343
  on HELPDEF TAG 350
  on KEYL TAG 362
  on PANDEF TAG 412
  on PANEL TAG 418
APPTITLE attribute
  on HELP TAG 343
  on HELPDEF TAG 350
  on PANDEF tag 412
  on PANEL tag 420
AREA (area) tag
  conditions of usage 226
  defining panel text portion 43
  description 225

© Copyright IBM Corp. 1989, 2001

AREA (area) tag *(continued)*
examples 43, 227
syntax 223
area divider (DIVIDER) tag
conditions of usage 297
description 297
examples 297
syntax 296
area divider (DLDIV) tag
conditions of usage 303
description 303
examples 303
ASIS attribute
on PANEL tag 421
ASSIGNI (assignment list item) tag
conditions of usage 230
description 230
examples 230
syntax 229
ASSIGNL (assignment list) tag
conditions of usage 231
description 231
examples 231
syntax 231
assignment list (ASSIGNL) tag
conditions of usage 231
description 231
examples 231
syntax 231
assignment list item (ASSIGNI) tag
conditions of usage 230
description 230
examples 230
syntax 229
ATTENTION (attention) tag
conditions of usage 233
description 232
syntax 232
attention message
description
format 142
ATTENTION tag
description 142
ATTN attribute
on ATTR tag 237
ATTR (attribute) tag
conditions of usage 237
description 237
examples 237
syntax 234
ATTRCHANGE attribute
description 104
data field 92
list field 110
on CHOFLD tag 258
on DTACOL tag 309
on DTAFLD tag 318
on LSTCOL tag 377
on LSTFLD tag 384
using 92, 110
ATTRCHAR attribute
description
data field 92
on ATTR tag 235
on DTAFLD tag 320
using 92
attribute
description 11
tag 11
attribute (ATTR) tag
conditions of usage 237
description 237
examples 237
syntax 234
attributes and values, coding 11
ATTRUSE attribute
on PANEL tag 420
AUTODMEM attribute
description
data field 93
on DTAFLD tag 320
using 93
AUTONRET attribute
on PANEL tag 422
AUTOSEL attribute
on CHOICE tag 266
AUTOTAB 102
AUTOTAB attribute
description 104, 110
on CHOFLD tag 256
on CHOICE tag 264
on CMDAREA tag 276
on DTACOL tag 308
on DTAFLD tag 316
on LSTCOL tag 375
on SELFLD tag 460
using 110
AUTOTCMD attribute
on PANEL tag 422
AUTOTYPE attribute
description
data field 93
on DTAFLD tag 320
using 93
AUTOVOL attribute
description
data field 93
on DTAFLD tag 320
using 93

B

BARRIER attribute on ACTION tag 220
BIT test
description 78
example 78
specifying on CHECKI tag 248
BMARGIN attribute
on PANDEF TAG 413
on PANEL TAG 422
BOTINST (bottom instruction) tag
conditions of usage 238
defining instruction text 42
description 238
examples 42, 239
syntax 238
bottom instruction
defining 42
description 6
bottom instruction (BOTINST) tag
conditions of usage 238
description 238
examples 239
bottom instruction (BOTINST) tag *(continued)*
syntax 238
BREAK=ALL attribute in definition list 133
BREAK attribute
on DL tag 299
on PARML tag 426
BREAK=FIT attribute in definition list 134

C

CAPS attribute
description 104
data field 93
input-only 112
input/output 112
on ATTR tag 236
on CMDAREA tag 277
on DTACOL tag 309
on DTAFLD tag 320
on LSTCOL tag 378
using 93
CASE attribute on KEYI tag 359
CAUTION (caution) tag
conditions of usage 240
description 143, 240
example 143
examples 241
syntax 240
CCSID attribute
on HELP tag 343, 398
on HELPDEF tag 350
on PANDEF tag 412
on PANEL tag 419
CDATA keyword
on ENTITY statement 202
character variables 63
CHARS test
specifying on CHECKI tag 246
CHDIV (choice divider) tag
conditions of usage 242
description 242
examples
solid and blank 242
syntax 242
check list
example 162
specifying message for 162
CHECKI (validity check item) tag
conditions of usage 252
description 252
examples 252
syntax 244
checking values within a numeric range, range test 71
CHECKL (validity check list) tag
conditions of usage 253
description 253
examples 253
syntax 253
using MSG attribute 161
CHECKVAR attribute
description 40
example 40
on CHOICE tag 263

CHECKVAR attribute *(continued)*
on PDC tag 431
CHKBOX attribute
description 103
on SELFLD tag 462
using 103
CHOFLD (choice data field) tag
conditions of usage 259
description 259
examples 260
syntax 254
CHOFLD (choice field) tag
using MSG attribute 161
choice 94, 100
CHOICE (selection choice) tag
conditions of usage 267
defining a selection field 94
description 266
examples 267
syntax 261
choice data field (CHOFLD) tag
conditions of usage 259
description 259
examples 260
syntax 254
choice divider (CHDIV) tag
conditions of usage 242
description 242
examples 242
syntax 242
CHOICECOLS attribute
description 102
on SELFLD tag 461
using 102
CHOICEDEPTH attribute
description 102
on SELFLD tag 461
using 102
CKBOX attribute
on ATTR tag 237
CLEAR attribute
description 110
on LSTCOL tag 375
CMD (command definition) tag
assigning function key 171
conditions of usage 269
description 269
examples 270
syntax 268
using to define a command 165
CMD attribute on KEYI tag 359
CMDACT (command action) tag
conditions of usage 272
description 272
examples 272
syntax 270
CMDAREA (command area) tag
conditions of usage 278
defining a command area 55
description 278
examples
defining 55
in an application panel markup 279
specifying the command prompt text 56
syntax 273
CMDLEN attribute
on CMDAREA tag 276
CMDLINE attribute
on PANEL tag 420
CMDLOC attribute
on CMDAREA tag 276
CMDTBL (command table) tag
assigning function key 171
conditions of usage 281
description 280
examples 281
syntax 280
using to define a command table 165
coding an action bar definition 37
COLOR attribute
description 104, 111
data field 92
on ATTR tag 236
on DTAFLD tag 320
on HP tag 355
on LSTCOL tag 377
on SELFLD tag 465
using 92, 104
COLSPACE attribute
description 111
on LSTCOL tag 377
COLTYPE attribute
description 111
on LSTCOL tag 377
column, defining widths 104, 110
column data 104
COLWIDTH attribute on LSTCOL tag 375
command 165
command action (CMDACT) tag
conditions of usage 272
description 272
examples 272
syntax 270
using to define a command action 167
command area
defining 55
defining cursor placement 33
description 7
command area (CMDAREA) tag
conditions of usage 278
description 278
examples 279
syntax 273
command definition (CMD) tag
conditions of usage 269
description 269
examples 270
syntax 268
command-prompt-text attribute
on CMDAREA tag 277
command table 165
command table (CMDTBL) tag
conditions of usage 281
description 280
examples 281
syntax 280
commands
application command table 165
declaring variables for 61
defining a command area 55
commands *(continued)*
defining application command table 165
defining with CMDTBL tag 165
description of application command table 9
specifying command action 167
specifying with the RUN attribute 38
truncating 168
comment (COMMENT) tag
examples 283
syntax 282
COMMENT (comment) tag
examples 283
syntax 282
comment delimiter
description 17
example 17
using 17
comments
examples 17
including in generated panel 15
including in source files 16
COMPACT attribute
on BOTINST tag 238
on DL tag 299
on GRPHDR tag 338
on NOTEL tag 402
on OL tag 406
on P tag 408
on PARML tag 426
on PNLINST tag 437
on SL tag 470
on TOPINST tag 478
on UL tag 480
using in a note list 402
using in a simple list 125
using in an ordered list 130
using in an unordered list 127
compact for simple list 125
compact lists
note 124
ordered 130
simple 125
unordered 127
compiler options (COMPOPT) tag
syntax 284
COMPOPT (compiler options) tag
examples 285
syntax 284
compopt (COMPOPT) tag
examples 285
considerations, compatibility for AREA tag 226
conversion utility
ACTBAR option 187
converting multiple source files 182
CUAATTR option 185
CUASUPP option 185
DBALIGN option 188
DBCS option 184
DISK option 184
DISPLAY option 187
DISPLAYW option 187
DSNCHK option 188
FORMAT option 186
GRAPHIC option 188

conversion utility *(continued)*
GUI option 187
help 182
installing 194
invocation panel input fields 178
invocation panel options 179
invocation panels 175
KANA option 185
KEYAPPL option 185
LISTING option 186
LISTREPL option 186
LOGREPL option 186
LSTVIEW option 185
MCOMMENT option 189
MERGESAREA option 187
messages 193
MSGEXPAND option 186
MSGSUPP option 185
NOACTBAR option 187
NOCUAATTR option 185
NOCUASUPP option 185
NODBALIGN option 188
NODBCS option 184
NODISPLAY option 187
NODISPLAYW option 187
NODSNCHK option 188
NOFORMAT option 186
NOGRAPHIC option 188
NOGUI option 187
NOKANA option 185
NOLISTING option 186
NOLISTREPL option 186
NOLOGREPL option 186
NOLSTVIEW option 185
NOMCOMMENT option 189
NOMERGESAREA option 187
NOMSGEXPAND option 186
NOMSGSUPP option 185
NOPANEL option 185
NOPLEB option 189
NOPREP option 185
NOREPLACE option 184
NOSCRIPT option 186
NOSTATS option 186
NOV3PADC option 189
NOVERSION option 187
NOZVARS option 188
overview 9
PANEL option 185
PLEB option 189
PREP option 185
PROFDDN option 189
PROFILE option 189
REPLACE option 184
SCREEN option 184
SCRIPT option 186
STATS option 186
supporting keys help 173
syntax 183
using 175
V3PADC option 189
VERSION option 187
ZVARS option 188
converting DTL source files 175
COPYR (copyright) tag
examples 286
syntax 286
copyright
including in generated panel 15
copyright (COPYR) tag
examples 286
copyright (COPYRIGHT) tag
syntax 286
CSRGRP attribute
description 103, 111
data field 91
on ATTR tag 237
on DTAFLD tag 318
on LSTCOL tag 377
on PS tag 440
on SELFLD tag 463
using 91, 103
CSRINDEX attribute
example 34
on PANEL tag 419
CSRPOS attribute
example 34
on PANEL tag 419
CUAATTR option 185
CUADYN attribute
on ATTR tag 237
CUASUPP option 185
CURSOR attribute
example 33
on PANEL tag 419
cursor field
on PANEL tag 419
cursor placement
ABC 32
characteristics 32
CHOICE 32
DTAFLD 32
in command area 32
in data field 32
in list field 32
in selection field 32
LSTCOL 32
SELFLD 32
CWIDTHS attribute
CWIDTHS attribute on SELFLD tag 461
description 102
using 102

D

DA (dynamic area) tag
conditions of usage 291
description 291
examples 292
syntax 287
using 49
data column
defining 104
example 104
data column (DTACOL) tag
conditions of usage 310
description 104, 310
examples 311
syntax 306
data field
attributes 90
defining a field prompt 83
defining alignment of data 90
data field *(continued)*
defining an associated message 90, 101
defining cursor placement 32
defining data columns 104
defining help for 89
defining input/output 86
defining width 88
examples 86
providing descriptive text 88
tailoring 90, 101
data field (DTAFLD) tag
conditions of usage 321
description 321
examples 321
syntax 312
data field description (DTAFLDD) tag
conditions of usage 323
description 88, 323
example 88
examples 323
syntax 323
DATAMOD attribute
on DA tag 289
DATAVAR attribute
description 62
on CHOFLD tag 255
on DTAFLD tag 314
on LSTCOL tag 374
on LSTVAR tag 389
DBALIGN attribute
description 103, 104
data field 92
on DTAFLD tag 319
on SELFLD tag 464
using 92, 103
DBALIGN option 188
DBCS
option 184
restrictions for leading and trailing blanks 13
DBCS test
specifying on CHECKI tag 249
test
description 75
example 75
DD (definition description) tag
conditions of usage 292
description 131, 292
examples
basic 131
BREAK=ALL attribute 133
BREAK=FIT attribute 134
help panel markup 293
syntax 292
DDHD (definition description header) tag
conditions of usage 294
description 131, 294
examples 134, 295
syntax 294
declaring variables 61
default keylist, key mappings 172
defining help for key list 173
definition description (DD) tag
conditions of usage 292
description 292

definition description (DD) tag
(continued)
examples
basic 131
BREAK=ALL attribute 133
BREAK=FIT attribute 134
help panel markup 293
syntax 292
definition description header (DDHD) tag
conditions of usage 294
description 294
examples 295
syntax 294
definition list (DL) tag
conditions of usage 300
description 299
examples
basic 131
BREAK=ALL attribute 133
BREAK=FIT attribute 134
default BREAK value of NONE 300
syntax 298
using to define a definition list 131
definition term (DT) tag
conditions of usage 305
description 305
examples
basic 131
BREAK=ALL attribute 133
BREAK=FIT attribute 134
help panel markup 305
syntax 304
definition term divider (DTDIV) tag
conditions 324
description 324
syntax 324
definition term header (DTHD) tag
conditions of usage 326
description 326
examples 327
syntax 326
definition term header divider (DTHDIV) tag
conditions 328
description 328
syntax 328
delimiter symbol 11
depth and width, defining with PANDEF tag 57
DEPTH attribute
defining application panel depth 31
description 102
data field 92
on AREA tag 224
on DA tag 289
on DTAFLD tag 318
on GA tag 334
on HELP tag 342
on HELPDEF tag 350
on PANDEF tag 411
on PANEL tag 417
on PS tag 440
on REGION tag 446
on SELFLD tag 460
using 92, 102
DESSKIP attribute
description
data field 92
on DTAFLD tag 319
using 92
DESTVAR attribute, on ASSIGNL tag 231
DESWIDTH attribute
description 104
on DTACOL tag 307
on DTAFLD tag 316
dialog element
creating 11
help panel 7
dialog elements
application command table 9
description 5
description of application panel 5
description of help panel 7
key mapping list 9
messages 8
types 5
variable classes 9
variables 9
Dialog Tag Language (DTL)
advantages 3
coding attributes and values 11
coding tag text 12
comments 15, 16
copyright statements 15
delimiters 11
document type declaration 16
entities 18
introduction 3
nesting tags 15
parameter entities 22
predefined symbols 25
relationship to CUA 4
results of converting 193
return codes 193
similarity to BookMaster 4
source file 11
syntax conventions 11
using 11
dialog variable 61
DIR attribute
on AREA tag 225
on REGION tag 446
DISK option 184
DISP attribute
on MSG tag 395
DISPLAY attribute
description 91
input-only 113
input/output 113
on CHOFLD tag 257
on DTAFLD tag 317
on LSTCOL tag 378
using 91
DISPLAY option 187
DISPLAYW option 187
DIV attribute
on AREA tag 224
on DA tag 289
on GA tag 334
on GRPHDR tag 338
on LSTFLD tag 383
DIVEND attribute
on DL tag 299
on PARML tag 426
DIVIDER (area divider) tag
conditions of usage 297
defining a divider 50
description 297
examples
solid and blank 54, 297
TYPE attribute 50
within a horizontal REGION tag 54
within vertical REGION tag 52
syntax 296
within a REGION tag 53
DIVLOC attribute
on GRPHDR tag 338
DIVWIDTH attribute
on AREA tag 225
DL (definition list) tag
conditions of usage 300
description 131, 299
examples
basic 131
BREAK=ALL attribute 133
BREAK=FIT attribute 134
default BREAK value of NONE 300
syntax 298
using to define a definition list 131
DLDIV (area divider) tag
conditions of usage 303
description 303
examples
solid and blank 303
DLDIV tag
syntax 302
DM application, providing help for 149
DOCTYPE statement
declaring document type 16
description 201
document type declaration 16
parameters 201
syntax 201
document type declaration
description 201
types supported by DTL 16
double-byte characters, permitting usage through variables 63
DSNAME test
description 73
example 73
specifying on CHECKI tag 250
DSNAMEF test
description 73
example 73
specifying on CHECKI tag 250
DSNAMEFM test
description 73
example 73
specifying on CHECKI tag 250
DSNAMEPQ test
description 74
example 74
specifying on CHECKI tag 250
DSNAMEQ test
description 74

DSNAMEQ test *(continued)*
example 74
specifying on CHECKI tag 250
DSNCHK option 188
DT (definition term) tag
conditions of usage 305
description 131, 305
examples
basic 131
BREAK=ALL attribute 133
BREAK=FIT attribute 134
help panel markup 305
syntax 304
DTACOL (data column) tag
conditions of usage 310
description 310
examples 311
syntax 306
DTAFLD (data field) tag 88
conditions of usage 321
description 321
examples 321
syntax 312
using 86
using MSG attribute 161
DTAFLDD (data field description) tag 88
conditions of usage 323
description 323
examples 323
syntax 323
DTDIV (definition term divider) tag
conditions 324
description 324
syntax 324
DTHD (definition term header) tag
conditions of usage 326
description 131, 326
examples 134, 327
syntax 326
DTHDIV (definition term header divider) tag
conditions 328
description 328
syntax 328
DTL
macros 207
results of converting 193
return codes 193
source files, converting 175
dynamic area (DA) tag
conditions of usage 291
description 291
examples 292
syntax 287
using 49

E

EBCDIC test
description 75
example 75
specifying on CHECKI tag 250
emphasizing panel text
description 144
HP (Highlighted phrase) 144
restriction 144
emphasizing panel text *(continued)*
RP (reference phrase) 144
end tag delimiters 11
ENDATTR attribute
on PANEL tag 421
entity
defining 18
description 18
examples
changing text 19
declaring a different name 22
declaring a file 20
declaring name and text string 18
naming conventions 19
specifying a name in text 19
using a text string in source file 21
using to imbed external files 23
naming conventions 18
parameter 22
predefined 25
using to imbed external files 23
entity declarations
conditions of usage 204
description 202, 203
example 204
entity definitions
example 205
ENTITY keyword
on ENTITY statement 202
entity-name
on ENTITY statement 202
ENTITY statement
parameters 202
syntax 202
entity-text
on ENTITY statement 203
ENTKEYTEXT attribute
on PANDEF TAG 412
on PANEL TAG 421
ENTWIDTH attribute
description 101, 104
on CHOFLD tag 257
on CMDAREA tag 275
on DTACOL tag 307
on DTAFLD tag 316
on SELFLD tag 459
using 101
ENUM 78
ENUM test
specifying on CHECKI tag 250
error messages 193
ERRORCHECK attribute
on PANEL tag 422
example (XMP) tag
conditions of usage 498
description 497
examples 120, 498
syntax 497
using to define an example 120
EXPAND attribute
description
data field 92
on CHOFLD tag 258
on DTAFLD tag 318
on HELP TAG 343
on HELPDEF TAG 350
EXPAND attribute *(continued)*
on PANDEF TAG 412
on PANEL tag 420
using 92
EXTEND attribute
description 102
on AREA tag 224
on DA tag 288
on GA tag 334
on REGION tag 446
on SELFLD tag 460
using 102

F

FCHOICE attribute
description 101
on SELFLD tag 460
using 101
field
defining 83
defining a data field message 90, 101
defining a list field message 111
defining data field 86
defining data field columns 104
defining list field 106
defining multiple-choice selection fields 95
defining selection fields 93
defining single-choice fields 94
interactive 83
providing additional information 148
types 83
field-level help
on LSTCOL tag 374
on SELFLD tag 457
field prompt
attributes 83
defining 83
example 83
specifying width 85
FIG (figure) tag
conditions of usage 330
description 330
examples 121, 331
syntax 329
using to define a figure 121
FIGCAP (figure caption) tag
conditions of usage 332
description 332
examples 122, 332
syntax 331
figure (FIG) tag
conditions of usage 330
description 330
examples 121, 331
syntax 329
using to define a figure 121
figure caption (FIGCAP) tag
conditions of usage 332
description 332
examples 122, 332
syntax 331
FileID test 73
FILEID test
specifying on CHECKI tag 251
files for installing the product 194

filespec
on ENTITY statement 203
FKA attribute on KEYI tag 359
FLDSPACE attribute
description 91, 104
on CHOFLD tag 257
on DTACOL tag 308
on DTAFLD tag 316
using 91
FLDTYPE attribute
description 104
data field 92
on DTAFLD tag 320
on SELFLD tag 464
using 92, 104
FLDWIDTH attribute
description
data field 92
on DTAFLD tag 318
using 92
FMTWIDTH attribute
on GRPHDR tag 338
FORMAT attribute
description 111
on AREA tag 225
on ATTR tag 236
on CHDIV tag 242
on DA tag 290
on DIVIDER tag 297, 303
on DL tag 299
on GRPHDR tag 337
on LSTCOL tag 375
on MSG tag 395
on PARML tag 426
on PLDIV tag 436
on XLATL tag 495
FORMAT option 186
formatting panel text
Asian rules 13
English rules 13
FRAME attribute on FIG tag 329
function key area
defining 171
description 7

G

GA (graphic area) tag
conditions of usage 335
description 334
examples 335
syntax 333
using 50
GAP attribute
on DIVIDER tag 296
on DLDIV tag 302
on PLDIV tag 436
GE attribute
on ATTR tag 236
generate (GENERATE) tag
conditions of usage 336
description 336
examples 336
syntax 335
GENERATE (generate) tag
conditions of usage 336
description 336
GENERATE (generate) tag *(continued)*
syntax 335
GENERATE (generate)tag
examples 336
generated panel comments
comments 15
generated panel statements
copyright 15
graphic area (GA) tag
conditions of usage 335
description 334
examples 335
syntax 333
using 50
GRAPHIC option 188
group header (GRPHDR) tag
conditions of usage 339
description 338
examples 339
syntax 337
GRPBOX attribute on REGION tag 447
GRPBXMAT attribute on REGION tag 447
GRPBXVAR attribute on REGION tag 447
GRPHDR (group header) tag
additional attributes for 113
conditions of usage 339
description 338
description of attributes 113
examples 339
syntax 337
GRPWIDTH attribute on REGION tag 447
GUI option 187
GUTTER attribute
on CHDIV tag 242
on DIVIDER tag 296
on DLDIV tag 303
on PLDIV tag 436

H

heading
in the information region 118
levels 118
heading (Hn) tag
conditions of usage 353
description 352
examples 118, 353
syntax 352
using to define a heading 118
HEADLINE attribute
on GRPHDR tag 338
on LSTGRP tag 386
help
defining a help pull-down 39
defining for a data field 89
defining help panels 149
field-level 457
for action bar 39
for application panels 32
for selection choice 99
for selection field 99
for the conversion utility 182
HELP (help panel) tag
conditions of usage 346
HELP (help panel) tag *(continued)*
description 345
examples 347
syntax 340
HELP attribute
on ABC tag 214
on CHOFLD tag 256
on CHOICE tag 262
on CMDAREA tag 275
on DA tag 291
on DTAFLD tag 315
on HELP tag 342
on HELPDEF tag 350
on KEYL tag 362
on LSTCOL tag 374
on MSG tag 394
on PANDEF tag 411
on PANEL tag 417
on PDC tag 430
on RP tag 453
on SELFLD tag 457
help default (HELPDEF) tag
conditions of usage 351
description 351
examples 352
syntax 349
help panel
defining for a data field 89
defining help panel text 150
function key area 8
layout 7
overview 7
panel body 8
panel title 7
title 7
types of help 7
using HELP attribute of PANEL tag 32
help panel (HELP) tag
conditions of usage 346
description 345
examples 347
syntax 340
help panel tag, using 149
help panels
defining 149
defining areas and regions 150
using the INFO tag with 150
help pull-down, defining 39
HELP tag 149
HELPDEF (help default) tag
conditions of usage 351
description 351
examples 352
syntax 349
HELPDEF attribute on HELP tag 342
HEX test
description 78
example 78
specifying on CHECKI tag 251
HIDE attribute on CHOICE tag 265
HIDEX attribute on CHOICE tag 265
highlighted phrase (HP) tag 454
conditions of usage 355
description 355
examples 355
syntax 354

HILITE attribute
description 104, 111
data field 92
on ATTR tag 236
on DTAFLD tag 320
on HP tag 355
on LSTCOL tag 378
on SELFLD tag 465
using 92, 104
Hn (heading) tag
conditions of usage 353
description 352
examples 118, 353
syntax 352
using to define a heading 118
horizontal region 53
HP (Highlighted phrase) tag
conditions of usage 355
description 144, 355
examples 144, 355
restriction 144
syntax 354

I

ID attribute
on HELPDEF tag 350
on PANDEF tag 411
ID panel 30
IDATE attribute
value IDATE on CHECKI tag 251
IDATE test
description 79
example 79
IMAPCOL attribute
on HELP tag 344
on HELPDEF tag 351
on PANDEF tag 413
on PANEL tag 422
IMAPNAME attribute
description
data field 92
on CHOFLD tag 258
on CMDAREA tag 277
on DTAFLD tag 319
on HELP tag 344
on HELPDEF tag 351
on PANDEF tag 413
on PANEL tag 422
on PS tag 440
using 92
IMAPNAMEP attribute
description
data field 92
on CMDAREA tag 277
on DTAFLD tag 319
on PS tag 441
using 92
IMAPROW attribute
on HELP tag 344
on HELPDEF tag 351
on PANDEF tag 413
on PANEL tag 422
imbedding source files 23
immediate-action, for pull-down choice 38
INCLUDE test
description 79
example 79
specifying on CHECKI tag 251
INDENT attribute
on AREA tag 224
on DL tag 299
on GRPHDR tag 338
on OL tag 406
on P tag 408
on PARML tag 426
on REGION tag 446
on SL tag 470
on UL tag 480
index value
on PANEL tag 419
INFO (information region) tag
conditions of usage 357
description 357
examples 115, 358
syntax 356
using to define an information region 115
information message
defining 160
description 160
providing for fields 148
information region
defining 115
defining text 116
definition lists 131
examples (XMP tag) 120
figure captions 122
figures 121
headings 118
list part 137
ordered lists 128
paragraphs 116
parameter lists 135
providing information for fields 148
simple lists 124
tags for text 115
unformatted text 119
unordered lists 126
information region (INFO) tag
conditions of usage 357
description 357
examples 115, 358
syntax 356
using to define an information region 115
INIT attribute
description 103
data field 92
on CHOFLD tag 258
on DTAFLD tag 318
on SELFLD tag 462
using 92, 103
initialization syntax, source-filespec 184
input/output data field 86
installing the conversion utility 194
instruction text 42
instructions, top and bottom 42
INTENS attribute
description 104, 111
data field 92
on ATTR tag 236
INTENS attribute *(continued)*
on DTAFLD tag 320
on HP tag 355
on LSTCOL tag 378
on SELFLD tag 465
using 92, 104
INTENSE attribute
on HP tag 355
on P tag 408
IPADDR4
description 80
example 80
IPADDR4 attribute
value IPADDR4 on CHECKI tag 252
IPADDR4 test
specifying on CHECKI tag 252
ISP application identifier 165
ISPCMDTB 165, 280
ISPDTLC
overview 9
using ? 183
using command 175
ISPKYLST
description 172
key mappings list 172
using 172
ISPx application identifier 165
item translate list
description 65
example 66
ITIME attribute
value ITIME on CHECKI tag 251
ITIME test
description 80
example 80

J

JDATE attribute
value JDATE on CHECKI tag 251
JDATE test
description 79
example 79
JSTD attribute
value JSTD on CHECKI tag 251
JSTD test
description 80
example 80
JUST attribute
on ATTR tag 236

K

KANA option 185
KEY attribute on KEYI tag 359
key item (KEYI) tag
conditions of usage 360
description 360
examples 360
syntax 358
using to define a key item 171
key list, defining help 173
key mapping list
defining 171
defining with PANDEF tag 57
overview 9

key mapping list *(continued)*
using KEYLIST attribute of PANEL tag 31
key mapping list (KEYL) tag
conditions of usage 363
description 362
examples 363
syntax 361
KEYAPPL option 185
KEYI (key item) tag
conditions of usage 360
description 171, 360
examples 360
syntax 358
using to define a key item 171
KEYL (key mapping list) tag
conditions of usage 363
description 171, 362
examples 363
syntax 361
using 171
KEYLIST attribute
examples 31
on HELP TAG 343
on HELPDEF TAG 350
on PANDEF tag 411
on PANEL tag 418
KEYLTYPE attribute
on HELP TAG 343
on HELPDEF tag 350
on PANDEF tag 412
on PANEL tag 418
keys 171
assigning actions 171
default keylist 172
defining 172
defining key mapping lists 171
displaying 172

L

LANG attribute on ACTION tag 220
LEN test
description 77
example 77
specifying on CHECKI tag 250
LI (list item) tag
conditions of usage 365
description 364
examples
basic unordered list 126
help panel with unordered list 366
list part (LP) tag 137
nested unordered 128
nesting ordered list 128
note list 124
paragraph nested in a list 136
simple list 124
syntax 364
LINE attribute
description 111
on LSTCOL tag 375
on LSTVAR tag 389
LINES (lines) tag
conditions of usage 367
description 367
LINES (lines) tag *(continued)*
examples 119, 367
syntax 366
using to define unformatted text 119
list column 34
defining width 110
truncating 110
list column (LSTCOL) tag
conditions of usage 379
description 378
examples 379
syntax 372
list field
additional attributes for 110
auto-tab attribute 110
defining 106
defining alignment of data 110
defining an associated message 111
defining cursor placement 32
defining required input for 112
description 106
example 107
specifying help for 111
tailoring 110
list field (LSTFLD) tag
conditions of usage 384
description 384
examples 385
syntax 382
list group (LSTGRP) tag
conditions of usage 387
description 387
example 388
syntax 386
list item (LI) tag
conditions of usage 365
description 364
examples
basic 124
COMPACT attribute on simple list 125
compact ordered list 130
help panel with unordered list 366
syntax 364
list part (LP) tag
conditions of usage 370
description 370
examples 137, 371
syntax 370
using to define a list part 137
list variable (LSTVAR) tag
conditions of usage 390
description 390
example 390
syntax 389
LISTDEPTH attribute
description 103
on SELFLD tag 463
using 103
LISTING option 186
LISTREF attribute
description 103
on SELFLD tag 463
using 103
LISTREPL option 186
lists
definition 131
list part 137
nesting lists within lists 138
note 124
ordered 128
parameter 135
simple 124
types 123
unordered 126
LISTTYPE attribute
description 103
on SELFLD tag 463
using 103
LISTV test
description 77
example 77
specifying on CHECKI tag 249
LISTVX test
description 77
example 77
specifying on CHECKI tag 249
LIT (literal) tag
conditions of usage 369
description 369
examples 369
syntax 368
literal (LIT) tag
conditions of usage 369
description 369
examples 369
syntax 368
LMSG attribute
on PANEL tag 421
LOCATION attribute
on MSG tag 395
on REGION tag 447
LOGREPL option 186
LP (list part) tag
conditions of usage 370
description 370
examples 137, 371
syntax 370
using to define a list part 137
LSTCOL (list column) tag
additional attributes for 110
conditions of usage 379
description 106, 378
description of attributes 110
examples 379
syntax 372
using MSG attribute 161
LSTFLD (list field) tag
conditions of usage 384
description 106, 384
examples 385
syntax 382
LSTGRP (list group) tag
conditions of usage 387
description 106, 387
example 388
syntax 386
LSTVAR (list variable) tag
conditions of usage 390
description 106, 390
example 390
syntax 389

LSTVIEW option 185
LVLINE attribute
on DA tag 288
on GA tag 334

M

M (mnemonic) tag
conditions of usage 392
description 392
example 393
parameters 392
syntax 392
macros
DTL 207
MARGIND attribute on AREA tag 224
MARGINW attribute
example 43
on AREA tag 224
markup, coding 11
markup declarations
comments 16
defining entities and parameter entities 18
document type declaration 16
entity declarations 18
types supported by DTL 16
markup language
advantages 4
description 3
MATCH attribute
description 40
example 40
on CHOICE tag 264
on PDC tag 431
MCOMMENT option 189
MENU attribute
on PANEL tag 419
MERGESAREA attribute
on HELP TAG 344
on HELPDEF TAG 351
on PANDEF TAG 412
on PANEL tag 421
MERGESAREA option 187
message
declaring variables for 61
defining 159
defining for a data field 90, 101
defining for a list field 111
example 8
for check list 162
member 159
specifying a variable in text 164
specifying type 160
types 159
message (MSG) tag
conditions of usage 396
description 396
examples 160, 397
MSG SUFFIX attribute 160
syntax 393
using to define a message 159
message identifier 160
message member 159
message member (MSGMBR) tag
conditions of usage 398
description 398
message member (MSGMBR) tag *(continued)*
examples
basic 159
defining a message member 399
specifying type 160
syntax 397
using to define a message member 159
messages
assigning for check list 161
assigning for data field 161
assigning for failing specified translation 161
assigning for failing validity check 161
assigning for list column 161
conversion utility 193
defining 159
description 159
error 193
overview 8
warning 193
MIX test
description 75
example 75
specifying on CHECKI tag 251
MNEMGEN attribute on AB tag 212
mnemonic (M) tag
conditions of usage 392
description 392
example 393
parameters 392
syntax 392
mnemonic choice selection
from pull-downs and action bars 40
support of 40
MODE attribute on ACTION tag 220
MSG (message) tag
conditions of usage 396
description 396
examples 160, 397
MSG SUFFIX attribute 160
syntax 393
using to define a message 159
MSG attribute
on CHECKL tag 253
on DTAFLD tag 315
on LSTCOL tag 374
on SELFLD tag 460
on VARCLASS tag 484
on XLATL tag 495
using 90, 101
MSGEXPAND option 186
MSGLINE attribute
on HELP tag 344
on PANEL tag 420
MSGMBR (message member) tag
conditions of usage 398
description 398
examples
basic 159
defining a message member 399
specifying type 160
syntax 397
using to define a message member 159
MSGSUPP option 185
MSGTYPE attribute on MSG tag 394
multiple-choice selection field
defining 95
discussion 95
example 95
MVS naming conventions 194

N

NAME attribute
identifying variables 61
on CHOICE tag 262
on CMD tag 269
on CMDAREA tag 275
on DA tag 288
on DTAFLD tag 314
on GA tag 333
on HELP tag 341
on KEYL tag 362
on MSGMBR tag 398
on PANEL tag 416
on SELFLD tag 457
on VARCLASS tag 482
on VARDCL tag 486
rules for variable names 211
using with the PANEL tag 30
NAME test
description 74
example 74
specifying on CHECKI tag 248
NAMEF test
description 74
example 74
specifying on CHECKI tag 248
naming conventions for MVS 194
NEST attribute on ACTION tag 220
nesting 142
lists within lists 138
ordered lists 128, 130
simple list 125
tags within lists 136
unordered lists 127
NEWAPPL attribute on ACTION tag 219
NEWPOOL attribute on ACTION tag 219
NEWWINDOW attribute on ACTION tag 219
NOACTBAR option 187
NOCHECK attribute on ACTION tag 220
NOCUAATTR option 185
NOCUASUPP option 185
NODBALIGN option 188
NODBCS option 184
NODISPLAY option 187
NODISPLAYW option 187
NODSNCHK option 188
NOENDATTR attribute
description 91, 111
on CHOFLD tag 257
on DIVIDER tag 296
on DTAFLD tag 317
on LSTCOL tag 376
using 91
NOFORMAT option 186

NOGRAPHIC option 188
NOGUI option 187
NOINIT attribute
on CMDAREA tag 275
NOJUMP attribute
description
data field 93
on CMDAREA tag 277
on DTAFLD tag 320
using 93
NOKANA option 185
NOLISTING option 186
NOLISTREPL option 186
NOLOGREPL option 186
NOLSTVIEW option 185
NOMATCH attribute on CHOICE tag 264
NOMCOMMENT option 189
NOMERGESAREA option 187
NOMSGEXPAND option 186
NOMSGSUPP option 185
NOPANEL option 185
NOPLEB option 189
NOPREP option 185
NOREPLACE option 184
NOSCRIPT option 186
NOSEL attribute
description 103
on SELFLD tag 464
using 103
NOSKIP attribute
on DL tag 299
on FIG tag 330
on LINES tag 367
on NOTE tag 400
on NOTEL tag 402
on NT tag 404
on OL tag 406
on SL tag 470
on UL tag 480
on XMP tag 497
NOSTATS option 186
NOTE (note) tag
conditions of usage 401
description 400
examples 401
syntax 400
note (NT) tag
conditions of usage 404
description 404
examples 405
syntax 404
note list (NOTEL) tag
conditions of usage 402
description 402
examples 403
syntax 402
NOTE tag
description 139
example 140
NOTEL (note list) tag
conditions of usage 402
description 139, 402
examples 140, 403
syntax 402
NOV3PADC option 189
NOVERSION option 187
NOZVARS option 188
NT (note) tag
conditions of usage 404
description 139, 404
example 141
examples 405
syntax 404
NUM test
description 78
example 78
specifying on CHECKI tag 249
NUMERIC attribute
on ATTR tag 236
numeric variables
converting 64
description 64
uses 64

O

OL (ordered list) tag
conditions of usage 407
description 406
examples
COMPACT attribute 130
list part (LP) tag 137
nested with paragraph 407
nesting 128
paragraph nested in a list 136
syntax 405
using to define an ordered list 128
OPT attribute on ACTION tag 220
ordered list (OL) tag
conditions of usage 407
description 406
examples
COMPACT attribute 130
list part (LP) tag 137
nested with paragraph 407
nesting 128
paragraph nested in a list 136
syntax 405
using to define an ordered list 128
OUTLINE attribute
description
data field 91
DTACOL tag 104
LSTCOL tag 111
seletion field 102
on ATTR tag 236
on CHOFLD tag 258
on CHOICE tag 265
on CMDAREA tag 275
on DTACOL tag 308
on DTAFLD tag 317
on LSTCOL tag 377
on PANDEF tag 412
on PANEL tag 420
on SELFLD tag 462
using 91, 102
output data field 86
overriding variable classes 81

P

P (paragraph) tag
conditions of usage 409
description 408
examples
basic 116
defining information region width 409
formatting of 117
nested in an ordered list 136
syntax 408
using to define a paragraph 116
PAD attribute
description
data field 91
DTACOL tag 104
LSTCOL tag 111
selection field 102
on ATTR tag 236
on CHOFLD tag 257
on CHOICE tag 264
on CMDAREA tag 275
on DTACOL tag 308
on DTAFLD tag 317
on LSTCOL tag 377
on PANDEF tag 412
on PANEL tag 420
on SELFLD tag 462
using 91, 102
PADC attribute
description
data field 91
DTACOL tag 104
LSTCOL tag 111
selection field 102
on ATTR tag 236
on CHOFLD tag 257
on CHOICE tag 264
on CMDAREA tag 275
on DTACOL tag 308
on DTAFLD tag 317
on LSTCOL tag 377
on PANDEF tag 412
on PANEL tag 420
on SELFLD tag 462
using 91, 102
PANDEF (panel default) tag
conditions of usage 413
defining panel defaults 57
description 413
examples
overriding a value 59
referring to default definitions 414
shared panel dimensions only 58
shared panel values 58
syntax 410
PANDEF attribute
on PANEL tag 417
panel 7
declaring variables for 61
defining fields 83
defining with the PANEL tag 29
PANEL (panel) tag
conditions of usage 423
defining a panel ID 30
defining an application panel 29

PANEL (panel) tag *(continued)*
defining cursor placement 32
defining panel NAME value 30
defining the panel title 31
defining the panel width and depth 31
description 423
examples
CURSOR attribute 33
HELP attribute 32
KEYLIST attribute 31, 424
start and end tags 30
WIDTH and DEPTH attributes 31
specifying a key mapping list 31
specifying a KEYLIST attribute 31
specifying associated help panel 32
syntax 415
panel body 6
panel default (PANDEF) tag
conditions of usage 413
description 413
examples 414
syntax 410
panel defaults 57
panel instruction (PNLINST) tag
conditions of usage 438
description 438
examples 438
syntax 437
PANEL option 185
panel region, defining 51
panel title
defining 31
description 6
for help panels 7
panel-title-text attribute
on PANEL tag 423
PANELSTMT attribute
on PANEL tag 421
paragraph (P) tag
conditions of usage 409
description 408
examples
basic 116
defining information region width 409
formatting of 117
nested in an ordered list 136
syntax 408
using to define a paragraph 116
parameter description (PD) tag
conditions of usage 428
description 428
examples 135, 429
syntax 428
parameter entity
description 22
examples 22
naming conventions 23
syntax 202
parameter list (PARML) tag
conditions of usage 427
description 426
examples 427
syntax 425
using to define a parameter list 135
parameter list divider (PLDIV) tag
syntax 435
parameter term (PT) tag
conditions of usage 443
description 443
examples 135, 443
syntax 442
parameter term divider (PTDIV) tag
conditions of usage 444
description 444
examples 444
syntax 444
PARM attribute
on ACTION tag 218
on KEYI tag 359
parm list divider (PLDIV) tag
conditions of usage 436
description 436
examples 436
PARML (parameter list) tag
conditions of usage 427
description 426
examples 135, 427
syntax 425
using to define a parameter list 135
PAS attribute
description 112
data field 91
on ATTR tag 236
on CHOFLD tag 258
on DTAFLD tag 318
on LSTCOL tag 377
using 91
PASSLIB attribute on ACTION tag 219
PASSTHRU 167
PD (parameter description) tag
conditions of usage 428
description 428
examples 135, 429
syntax 428
PDC (pull-down choice) tag
conditions of usage 432
description 432
examples
basic 432
defining help panel 39
markup 38
using mnemonic selection 40
syntax 430
PDCVAR attribute on ABC tag 215
PDSEP (pull-down separator) tag
conditions of usage 434
description 434
example 434
syntax 434
phrase-to-be-highlighted attribute
on HP tag 355
PICT test
description 76
example 76
specifying on CHECKI tag 248
PICTCN test
description 76
example 76
specifying on CHECKI tag 248
PLACE attribute
description
data field 92
on CMDAREA tag 277
on DTAFLD tag 319
on PS tag 441
using 92
PLDIV (parm list divider) tag
conditions of usage 436
description 436
examples
solid and blank 436
syntax 435
PLEB option 189
PMTFMT attribute
description 104
data field 91
on DTACOL tag 308
on DTAFLD tag 317
using 91
PMTLOC attribute
description 104
on CMDAREA tag 275
on DTAFLD tag 317
on SELFLD tag 458
using 83
PMTSKIP attribute
description 103
data field 92
on DTAFLD tag 319
on SELFLD tag 464
using 92, 103
PMTTEXT attribute
on CMDAREA tag 275
PMTWIDTH attribute
description 104
on DTACOL tag 307
on DTAFLD tag 316
on SELFLD tag 458
using 83
PNLINST (panel instruction) tag
conditions of usage 438
description 438
examples 438
syntax 437
point-and-shoot (PS) tag
conditions of usage 441
description 441
example 441
syntax 439
pop-up window, displaying messages on 159
POSITION attribute
description 112
on LSTCOL tag 375
position value
on PANEL tag 419
predefined entities 25
predetermined tag attributes 12
PREP option 185
preselected pull-down choice 40
PRIME attribute
on PANEL tag 419
PROFDDN option 189
PROFILE option 189
prompt 83

prompt-width, specifying for data field 85
PS (point-and-shoot) tag
conditions of usage 441
description 441
example 441
syntax 439
PSBUTTON attribute
on CMDAREA tag 277
PSVAL attribute
description
data field 91
on CHOFLD tag 258
on CMDAREA tag 277
on DTAFLD tag 318
using 91
PSVAR attribute
description
data field 91
on CHOFLD tag 258
on CMDAREA tag 277
on DTAFLD tag 318
using 91
PT (parameter term) tag
conditions of usage 443
description 443
examples 135, 443
syntax 442
PTDIV (parameter term divider) tag
conditions of usage 444
description 444
examples 444
syntax 444
pull-down, defining 36
pull-down choice
(PDC) tag
conditions of usage 432
description 432
examples 432
syntax 430
actions 38
defining 36
example 39
preselected 40
providing help for 39
pull-down separator (PDSEP) tag
(PDSEP) tag
conditions of usage 434
description 434
example 434
syntax 434

R

RANGE test
checking values within a numeric range 71
description 71
specifying on CHECKI tag 245
reference phrase (RP) tag
conditions of usage 453
description 453
examples 454
syntax 452
REFRESH attribute
description 103
on SELFLD tag 462
REFRESH attribute *(continued)*
using 103
REGION (region) tag
conditions of usage 448
defining a region 51
description 448
examples
DIR attribute 51
horizontal and vertical 54, 449
syntax 445
REPLACE keyword
on ENTITY statement 202
REPLACE option 184
REQUIRED attribute
description 104
data field 90
LSTCOL tag 112
selection field 101
on CHOFLD tag 256
on DTACOL tag 309
on DTAFLD tag 315
on LSTCOL tag 374
on SELFLD tag 460
using 90, 101
RESULT attribute on ASSIGNI tag 230
return codes, results of converting with DTL 193
risk (attention statement) 142
risk (warning statement) 142
ROWS attribute
on LSTFLD tag 383
RP (Reference phrase) tag
conditions of usage 453
description 146, 453
example 146
examples 454
restriction 146
syntax 452
rules
for variable names 211
formatting Asian panel text 13
formatting English panel text 13
RULES attribute
on LSTFLD tag 383
RUN attribute
example 166
on ACTION tag 218
specifying a command 38, 166

S

SCRCAPS attribute
on CMDAREA tag 277
on DA tag 290
on LSTFLD tag 384
SCREEN option 184
SCRIPT option 186
SCRNAME attribute on ACTION tag 220
SCROLL attribute
on DA tag 289
SCROLLTAB attribute
on CMDAREA tag 276
on DA tag 290
on LSTFLD tag 384
SCROLLVAR attribute
on CMDAREA tag 276
SCROLLVAR attribute *(continued)*
on DA tag 290
on LSTFLD tag 383
SCRVHELP attribute
on CMDAREA tag 276
on DA tag 290
on LSTFLD tag 383
SELCHAR attribute on CHOICE tag 264
SELCHECK parameter 104
SELDEFAULT attribute
description 103
on SELFLD tag 464
using 103
selection choice (CHOICE) tag
conditions of usage 267
description 266
examples 267
syntax 261
selection choice, defining space for 100
selection field
attributes 101
defining 93
defining a field prompt 83
defining cursor placement 32
defining space for choice 100
help for 99
using the CHOICE tag 94
selection field (SELFLD) tag
conditions of usage 466
description 465
examples 467
syntax 455
selection list, defining cursor placement 34
selection width, defining 101
SELFLD (selection field) tag
conditions of usage 466
description 465
examples 467
syntax 455
using MSG attribute 162
SELFMT attribute
description 103
on SELFLD tag 462
using 103
SELMSG attribute
description 102
on SELFLD tag 462
using 102
SELMSGU attribute
description 102
on SELFLD tag 462
using 102
SELWIDTH attribute
defining space for 100
description 100, 104
on DTACOL tag 307
on SELFLD tag 458
using 100
SETVAR attribute on ACTION tag 221
SETVERB 167
SHADOW attribute
on DA tag 289
simple list (SL) tag
conditions of usage 470
description 470

simple list (SL) tag *(continued)*
examples
basic 124
compact and nested 471
COMPACT attribute 125
syntax 470
using to define a simple list 124
single-choice selection field
defining 94
discussion 94
example 94
SKIP attribute
on ATTR tag 236
on PARML tag 426
SL (simple list) tag
conditions of usage 470
description 470
examples
basic 124
compact and nested 471
COMPACT attribute 125
syntax 470
using to define a simple list 124
SMSG attribute
on MSG tag 396
on PANEL tag 421
SORT attribute on CMDTBL tag 280
source (SOURCE) tag
example 473
SOURCE (source) tag
example 473
SOURCE (Source) tag
conditions of usage 473
description 472
syntax 472
source file
defining dialog elements 11
defining entities and parameter entities 18
DOCTYPE declaration 201
frequently used words (entities) 18
imbedding 23
including comments 15, 16
including copright statements 15
source-filespec, for system 184
SPACE attribute
on LI tag 364
on OL tag 406
on SL tag 470
on UL tag 480
SPACE keyword
on ENTITY statement 202
specifying
a list of values to match value of user input 71
a list of valuesx to match value of user input 72
calling the conversion utility 184
source-filespec 184
start tag delimiters 11
STATS option 186
STDDATE attribute
value STDDATE on CHECKI tag 251
STDDATE test
description 79
example 79
STDTIME attribute
value STDTIME on CHECKI tag 252
STDTIME test
description 80
example 80
STRIP attribute
on GRPHDR tag 338
SUFFIX attribute on MSG tag 394
SUSPEND attribute on ACTION tag 219
syntax diagrams, reading 199
SYSTEM keyword
on ENTITY statement 203

T

T (truncation) tag
conditions of usage 474
description 474
examples 168, 474
syntax 474
using to define command truncation 168
tag
attributes 11
coding text 12
delimiters 11
description 11
nesting 15
rules for coding values 12
text 12
values 11
tags
AB 211
ABC 214
ACTION 217
ASSIGNI 229
ASSIGNL 231
ATTENTION 139, 232
ATTR 234
BOTINST 238
CAUTION 139
CHDIV 242
CHECKI 244
CHECKL 253
CHOFLD 254
CHOICE 261
CMD 268
CMDACT 270
CMDAREA 273
CMDTBL 280
COMMENT 282
COMPOPT 284
COPYRIGHT 286
DA 287
DD 292
DDHD 294
DIVIDER 296
DL (definition list) 298
DLDIV 302
DT 304
DTACOL 306
DTAFLD 312
DTAFLDD 323
DTDIV 324
DTHD 326
DTHDIV 328
FIG 329
tags *(continued)*
FIGCAP 331
for information region 115
GA 333
GENERATE 335
GRPHDR 337
HELP 340
HELPDEF 349
Hn 352
HP 354
INFO 356
KEYI 358
KEYL 361
LI 364
LINES 366
LIT 368
LP 370
LSTCOL 372
LSTFLD 382
LSTGRP 386
LSTVAR 389
M 392
mnemonic 392
MSG 393
MSGMBR 397
nesting 138
NOTE 400
NOTEL 402
NT 139, 404
OL 405
P 408
PANDEF 410
PANEL 415
PARML 425
PD 428
PDC 430
PDSEP 434
PLDIV 435
PNLINST 437
PS 439
PT 442
PTDIV 444
REGION 445
RP 452
SELFLD 455
SL 470
SOURCE 472
T 474
TEXTLINE 475
TEXTSEG 475
TOPINST 477
UL 479
VARCLASS 482
VARDCL 486
VARLIST 487
VARSUB 489
WARNING 139, 491
XLATI 493
XLATL 495
XMP 497
text
adding to a list 137
defining for information region 116
help 150
providing attention to user 142
providing caution 143
providing notes to user 139

text *(continued)*
providing warning to user 142
static 115
tag 12
tags used to define 115
using the example tag 120
using the figure tag 121
using the heading tags 118
using the LINES tag 119
using the paragraph tag 116
TEXT attribute
description 112
on AREA tag 225
on LSTCOL tag 376
on NOTE tag 400
on NOTEL tag 402
on NT tag 404
on OL tag 406
on SL tag 470
on UL tag 480
text line (TEXTLINE) tag
conditions of usage 475
description 475
examples 475
syntax 475
text segment (TEXTSEG) tag
conditions of usage 476
description 476
examples 476
syntax 475
TEXTFMT attribute
description 112
on LSTCOL tag 376
TEXTLEN attribute
description 112
on LSTCOL tag 376
TEXTLINE (text line) tag
conditions of usage 475
description 475
syntax 475
TEXTLINE (text line)tag
examples 475
TEXTLOC attribute
description 112
on LSTCOL tag 376
TEXTSEG (text segment) tag
conditions of usage 476
description 476
syntax 475
TEXTSEG(text segment)tag
examples 476
TEXTSKIP attribute
description 112
on LSTCOL tag 376
title, panel 31
TITLINE attribute
on PANEL tag 420
TMARGIN attribute
on PANDEF TAG 413
on PANEL TAG 422
TOGVAR attribute on ACTION tag 221
top instruction
defining 42
description 6
top instruction (TOPINST) tag
conditions of usage 478
description 478
top instruction (TOPINST) tag *(continued)*
examples 478
syntax 477
TOPINST (top instruction) tag
conditions of usage 478
defining instruction text 42
description 478
examples 42, 478
syntax 477
TRAIL attribute
description 102
on SELFLD tag 461
using 102
translate item (XLATI) tag
conditions of usage 494
description 494
examples 494
syntax 493
translate list (XLATL) tag
conditions of usage 496
description 496
examples 496
syntax 495
translate lists
defining 65
types 65
translation
defining items for 65
example 66, 67
TRUNC attribute
on CHOICE tag 265
on XLATL tag 495
truncation (T) tag
conditions of usage 474
description 474
examples 168, 474
syntax 474
using to define command truncation 168
TSIZE attribute
description 103
on DL tag 299
on PARML tag 426
on SELFLD tag 463
using 103
TUTOR attribute
on HELP tag 343
on PANEL tag 419
tutor-choice fields 99
TYPE attribute
description 93
on ACTION tag 219
on ATTR tag 235
on CHDIV tag 242
on CHECKI tag 245
on COMMENT tag 282
on DIVIDER tag 296
on DLDIV tag 302
on HP tag 355
on PANEL tag 421
on PLDIV tag 435
on SELFLD tag 458
on SOURCE tag 472
on VARCLASS tag 483
using 93
value %varname on VARCLASS tag 483
TYPE attribute *(continued)*
value ALPHA on CHECKI tag 246
value ALPHAB on CHECKI tag 250
value ANY on VARCLASS tag 483
value BIT on CHECKI tag 248
value CHAR on VARCLASS tag 483
value CHARS on CHECKI tag 246
value DBCS on CHECKI tag 249
value DBCS on VARCLASS tag 483
value DSNAME on CHECKI tag 250
value DSNAMEF on CHECKI tag 250
value DSNAMEFM on CHECKI tag 250
value DSNAMEPQ on CHECKI tag 250
value DSNAMEQ on CHECKI tag 250
value EBCDIC on CHECKI tag 250
value EBCDIC on VARCLASS tag 483
value ENUM on CHECKI tag 250
value FILEID on CHECKI tag 251
value HEX on CHECKI tag 251
value IDATE on VARCLASS tag 483
value INCLUDE on CHECKI tag 251
value IPADDR4 on CHECKI tag 252
value ITIME on VARCLASS tag 483
value JDATE on VARCLASS tag 484
value JSTD on VARCLASS tag 484
value LEN on CHECKI tag 250
value LISTV on CHECKI tag 249
value LISTVX on CHECKI tag 249
value MIX on CHECKI tag 251
value MIXED on VARCLASS tag 483
value NAME on CHECKI tag 248
value NAMEF on CHECKI tag 248
value NUM on CHECKI tag 249
value NUMERIC on VARCLASS tag 484
value PICT on CHECKI tag 248
value PICTCN on CHECKI tag 248
value RANGE on CHECKI tag 245
value STDDATE on VARCLASS tag 483
value STDTIME on VARCLASS tag 483
value VALUES on CHECKI tag 247
value VALUESX on CHECKI tag 247
value VMASK on VARCLASS tag 484

U

UL (unordered list) tag
conditions of usage 480
description 480
examples
basic 126
basic and nested 481
nested 128
syntax 479
using to define an unordered list 126
UNAVAIL attribute
on CHOICE tag 265
on PDC tag 431

UNAVAILMAT attribute on CHOICE tag 265
unformatted text
  using the example tag 120
  using the figure tag 121
  using the LINES tag 119
unordered list (UL) tag
  conditions of usage 480
  description 480
  examples
    basic 126
    basic and nested 481
    nested 128
  syntax 479
  using to define an unordered list 126
upper translate list
  example 65
  translating a value to uppercase 65
USAGE attribute
  description 106
  on CHOFLD tag 256
  on DTAFLD tag 315
  on LSTCOL tag 374
USERMOD attribute
  on DA tag 289

V

V3PADC option 189
validity check item (CHECKI) tag
  conditions of usage 252
  description 252
  examples 252
  syntax 244
validity check list (CHECKL) tag
  conditions of usage 253
  description 253
  examples 162, 253
  syntax 253
validity checks
  description 70
  types 70
  using 70
  using CHECKL and CHECKI tags 70
value
  description 11
  tag 11
VALUE attribute
  on ACTION tag 221
  on ASSIGNI tag 229
  on PS tag 440
  on XLATI tag 493
VALUE1 attribute on ACTION tag 221
VALUE2 attribute on ACTION tag 221
VALUES test
  description 71
  example 71
  specifying on CHECKI tag 247
VALUESX test
  description 72
  example 72
  specifying on CHECKI tag 247
VAR attribute
  on PS tag 440
  on VARSUB tag 489
VARCLASS (variable class) tag
  conditions of usage 485
VARCLASS (variable class) tag (continued)
  description 484
  examples 485
  syntax 482
  using MSG attribute 162
  using to define a variable class 63
VARCLASS attribute
  description 91, 104
  input-only 112
  input/output 112
  on CHOFLD tag 255
  on DTAFLD tag 315
  on LSTCOL tag 374
  on VARDCL tag 486
  using 91
VARDCL (variable declaration) tag
  conditions of usage 486
  description 486
  examples 62, 486
  syntax 486
VARDCL attribute
  description
    data field 93
  on CMDAREA tag 277
  on DTACOL tag 309
  on DTAFLD tag 320
  on LSTCOL tag 378
  on LSTFLD tag 384
  on SELFLD tag 465
  using 93
variable class
  associating translate list 65
  defining 62
  defining character variables 63
  defining translate list 495
  description 61
  overriding 81
  overriding the data variable 112
  overview 9
variable class (VARCLASS) tag
  conditions of usage 485
  description 484
  examples 485
  syntax 482
variable data
  aligning 90, 110
  description 110
variable declaration (VARDCL) tag
  conditions of usage 486
  description 486
  examples 486
  syntax 486
variable list
  defining 61
variable list (VARLIST) tag
  conditions of usage 487
  description 487
  examples 488
  syntax 487
variable names
  as a tag value, assigning 12
  rules 211
  rules for %varname as a tag value 211
variable substitution (VARSUB) tag
  conditions of usage 489
variable substitution (VARSUB) tag (continued)
  description 489
  examples 164, 490
  syntax 489
  using to substitute a variable 164
variable translate list
  defining 65
  relation to VARCLASS definition 65
  types 65
variable validation 65
variables
  character 63
  declaring 61
  defining a variable class 62
  in data fields 86
  item translating 66
  numeric 64
  overview 9
  rules for naming 211
  specifying in message text 164
  translate list 65
  translating 65, 66
  validating 65
  validity checks 70
VARLIST (variable list) tag
  conditions of usage 487
  description 487
  examples 62, 488
  syntax 487
  using 61
VARSUB (variable substitution) tag
  conditions of usage 489
  description 489
  examples 490
  syntax 489
VERIFY attribute
  description 103
  on SELFLD tag 462
  using 103
VERSION option 187
vertical region 52

W

WARNING (warning) tag
  conditions of usage 491
  description 491
  examples 233, 492
  syntax 491
warning message
  defining 160
  description
    format 142
    information messages 160
    ISPF conversion utility messages 193
  example 142
WARNING tag
  description 142
  example 142
width and depth, defining with PANDEF tag 57
WIDTH attribute
  defining application panel width 31
  on AREA tag 225
  on DA tag 289

WIDTH attribute *(continued)*
on FIG tag 329
on GA tag 334
on GRPHDR tag 338
on HELP tag 342
on HELPDEF tag 350
on INFO TAG 357
on MSGMBR tag 398
on PANDEF tag 411
on PANEL tag 418
on REGION tag 446
window
layout 5
WINDOW attribute
on PANDEF tag 412
on PANEL tag 419
WINTITLE attribute
on HELP TAG 343
on HELPDEF TAG 350
on PANDEF tag 412
on PANEL tag 420
word-wrapping
data field width 88
on DESWIDTH attribute 88
WSDIR attribute on ACTION tag 220
WSINVOKE attribute on ACTION tag 220
WSSIZE attribute on ACTION tag 220
WSVIEW attribute on ACTION tag 220

X

XLATI (translate item) tag
conditions of usage 494
description 494
examples 494
syntax 493
XLATL (translate list) tag
conditions of usage 496
description 496
examples 496
syntax 495
using MSG attribute 162
XMP (example) tag
conditions of usage 498
description 497
examples 120, 498
syntax 497
using to define an example 120

Z

ZCMD system variable
for PASSTHRU 167
for SETVERB 167
ZCONT attribute
on PANEL tag 422
ZGUI attribute
description 103
on SELFLD tag 463
using 103
ZKEYHELP, using 173
ZUP attribute
on PANEL tag 422
ZVARS option 188
ZVERB system variable for SETVERB 167

File Number: S370/4300-39
Program Number: 5694-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC34-4824-01